Ανοικτό Πανεπιστήμιο Κύπρου

Σχολή Θετικών και Εφαρμοσμένων Επιστημών

Μεταπτυχιακή Διατριβή
στα Πληροφοριακά και Επικοινωνιακά Συστήματα

Δημιουργία Συστήματος "Online Judge" Για Την Αυτόματη Αξιολόγηση Προγραμματιστικών Ασκήσεων

Παύλος Παυλικκάς

Επιβλέπων Καθηγητής
Μαρία Ανδρέου

Δεκέμβριος 2013

Ανοικτό Πανεπιστήμιο Κύπρου

Σχολή Θετικών και Εφαρμοσμένων Επιστημών

Δημιουργία Συστήματος "Online Judge" Για Την Αυτόματη Αξιολόγηση Προγραμματιστικών Ασκήσεων

Παύλος Παυλικκάς

Επιβλέπων Καθηγητής
Μαρία Ανδρέου

Η παρούσα μεταπτυχιακή διατριβή υποβλήθηκε
προς μερική εκπλήρωση των απαιτήσεων για απόκτηση

μεταπτυχιακού τίτλου σπουδών
στα Πληροφοριακά Συστήματα

από τη Σχολή Θετικών και Εφαρμοσμένων Επιστημών
του Ανοικτού Πανεπιστημίου Κύπρου

Δεκέμβριος 2013

Περίληψη

Κυριότερος στόχος του συστήματος που αναπτύχθηκε ήταν να προσφέρει αυτάρκεια στους καθηγητές και μαθητές που ασχολούνται με την Ολυμπιάδα Πληροφορικής όσο αφορά την προετοιμασία τους. Το σύστημα προσφέρει τη δυνατότητα στους καθηγητές να ανεβάσουν ασκήσεις, οι οποίες μπορούν να λυθούν από τους μαθητές και να αξιολογηθούν αυτόματα από το σύστημα. Επίσης έχουν δημιουργηθεί ασκήσεις με τα κατάλληλα test cases που θα βοηθήσουν τους μαθητές να καταλάβουν βασικές αλγοριθμικές τακτικές και μεθοδολογίες. Το σύστημα οργανώνει τις ασκήσεις με βάση τη θεματολογία καθώς και με βάση το επίπεδο δυσκολίας τους. Επιπλέον για κάθε θεματολογία αναπτύχθηκε σχετική θεωρία ώστε οι μαθητές να έχουν συγκεντρωμένο υλικό το οποίο θα πρέπει να διαβάσουν. Τόσο οι ασκήσεις όσο και η θεωρία θα μπορεί να εμπλουτίζεται αλλά και να διορθώνεται από τους καθηγητές. Στο σημείο αυτό θα ήθελα να τονίσω τη μεγάλη σημασία που έχει η σωστή επιλογή των test cases για κάθε άσκηση, ώστε να καλύπτονται όλες οι πιθανές περιπτώσεις. Πριν προχωρήσουμε στην ανάπτυξη του συστήματος έγινε επισκόπηση παρόμοιων συστημάτων και μελετήσαμε τις απόψεις διοργανωτών, καθηγητών και μαθητών που εμπλέκονται στις ολυμπιάδας πληροφορικής, τόσο στην Κύπρο όσο και στο εξωτερικό. Η δημιουργία του συστήματος ήταν μεγάλη πρόκληση τόσο σχεδιαστικά όσο και προγραμματιστικά. Η μεθοδολογία που ακολουθήθηκε ήταν η ακόλουθη:

- Προκαταρτική έρευνα. Εξακριβώθηκε η αναγκαιότητα του συστήματος.

- Εξακρίβωση των αναγκών και καθορισμός των απαιτήσεων. Έγινε με συνεντεύξεις, ερωτηματολόγια και μελέτη υπαρχόντων συστημάτων.

- Καταγραφή των προδιαγραφών. Έγινε μια τυποποιημένη καταγραφή των προδιαγραφών του συστήματος

- Σχεδιασμός του συστήματος. Η φάση αυτή περιλάμβανε τη σχεδίαση της βάσης δεδομένων, του interface και της αρχιτεκτονικής του συστήματος

- Υλοποίηση. Στην υλοποίηση εφαρμόσαμε όσα σχεδιάστηκαν στην προηγούμενη φάση έχοντας πάντα υπόψη τις προδιαγραφές που θέσαμε.

- Έλεγχος. Ετοιμάστηκε ένας κατάλογος ελέγχων που θα πρέπει να περάσει το σύστημα.

Summary

The main objective of the system was to provide self-sufficiency for teachers and students engaged in the Olympiad in Informatics for their preparation in the participation in international contests. The system provides the opportunity for teachers to raise exercises, which can be solved by students and be evaluated automatically. Also a variety of exercises has been created with the appropriate test cases that will help the students to understand basic algorithmic tactics and methodologies. The system organizes exercises by topics and by the level of difficulty. Furthermore, for each relevant topic some theory was developed which the students can study. Both exercises and theory can be enriched and corrected by teachers. At this point I would like to emphasize the great importance of proper selection of the test cases for each exercise, to cover all possible cases.

Before proceeding to the development of the system a study was made on similar systems. Also I asked for the opinions of organizers, teachers and students involved in the Olympiad in Informatics, both in Cyprus and abroad.

The creation of the system was a big challenge both on design and programming. The methodology used was as follows:

• Preliminary investigation.

- Determination of needs and definition of requirements. It was done throw interviews, questionnaires and the study of existing systems.

- Recording of specifications.

- System design. This phase included the design of the database, the interface and the system architecture.

- Implementation. In the implementation we applied those that had designed in the previous phase keeping in mind the standards we set.

- Checking based on checklist.

Ευχαριστίες

Θα ήθελα να εκφράσω τις ιδιαίτερες μου ευχαριστίες στην επιβλέπων καθηγήτρια μου Δρ. Μαρία Ανδρέου για την πολύτιμη της βοήθεια σε κάθε βήμα αυτής της πορείας.

Θα ήθελα να ευχαριστήσω επίσης τους μαθητές μου για τις πολύτιμες υποδείξεις τους.

Τέλος θα ήθελα να ευχαριστήσω τη γυναίκα μου Γιώτα για την αμέριστη συμπαράσταση που μου παρείχε όλον αυτόν τον καιρό.

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΙΝΑΚΑΣ ΣΧΗΜΑΤΩΝ

ΚΕΦΑΛΑΙΟ 1

Εισαγωγή

Ένα σύστημα online judge (ή auto grater) είναι ένα σύστημα που μπορεί να αξιολογήσει αυτόματα ένα πρόγραμμα βασιζόμενο στις εξόδους (outputs) που θα παράξει, το πρόγραμμα, για συγκεκριμένες εισόδους(inputs). Η υποβολή του προγράμματος γίνεται μέσω μιας σελίδας στο internet (συνήθως) εξού και το πρόθεμα online. Για κάθε πρόγραμμα, πέρα των αναμενόμενων εξόδων μπορούν να τεθούν και επιπλέον περιορισμοί που αφορούν τη μνήμη που χρησιμοποιεί το πρόγραμμα αλλά και το χρόνο εκτέλεσης. Τα συστήματα αυτά χρησιμοποιούνται κατά κύριο λόγο σε εθνικό επίπεδο για να προετοιμάσουν τους μαθητές για τους τοπικούς και διεθνείς διαγωνισμούς. Μπορούν να χρησιμοποιηθούν τόσο από πανεπιστήμια για την προετοιμασία των φοιτητών τους, όσο και από ανεξαρτήτους φορείς που διοργανώνουν διάφορους διαγωνισμούς. Ενδεικτικά στον πίνακα αναφέρονται μερικοί από τους πιο γνωστούς online judges.

Ονομασία	**Web site**	**Χρήση**
Usaco	http://train.usaco.org/usacogate	Προετοιμασία των μαθητών των ΗΠΑ
Hellenico	http://www.hellenico.gr/	Προετοιμασία των μαθητών της Ελλάδας
UVa	http://uva.onlinejudge.org/	Αναπτύχθηκε από το πανεπιστήμιο του Valladolid
SPOJ	http://www.spoj.pl/	«Πολυεθνικός». Ξεκίνησε από την Πολωνία
Topcoder	http://www.topcoder.com/	Διεθνείς Διαγωνισμοί

Η Κυπριακή Εταιρεία Πληροφορικής (Cyprus Computer Society – CCS), που είναι μαζί με το Υπουργείο Παιδείας ο διοργανωτής της Παγκύπριας Ολυμπιάδας Πληροφορικής δεν έχει στη διάθεση της ένα τέτοιο σύστημα εκπαίδευσης των μαθητών. Οι Κύπριοι μαθητές παραπέμπονται στα συστήματα άλλων χωρών. Το πρόβλημα με την απουσία εγχώριου συστήματος είναι ότι η κάθε χώρα έχει διαφορετικό επίπεδο εκπαίδευσης και συνήθως παραλείπουν τα αρχικά στάδια γιατί οι μαθητές αποκτούν αυτή τη γνώση στο σχολείο.

Το σύστημα μας φιλοδοξεί να καλύψει τους πιο κάτω στόχους:

- Να περιέχει ασκήσεις διαβαθμισμένες στις ανάγκες των Κυπρίων μαθητών.
- Να δίνει πρόσβαση στους μαθητές σε σχετική θεωρία για κάθε ενότητα.
- Να είναι εύχρηστο ώστε να μπορεί να γίνεται χρήση του και από άτομα μικρής ηλικίας.
- Να είναι δυναμικό παρέχοντας τη δυνατότητα στους καθηγητές να προσθέτουν, αφαιρούν και επεξεργάζονται ασκήσεις.

Οι φάσεις οι οποίες ακολουθήσαμε κατά τη διάρκεια ανάπτυξης του συστήματος ήταν [01]:

- Φάση Προκαταρκτικής Έρευνας. Στόχος αυτής της φάσης ήταν να αποφανθεί η αναγκαιότητα της ανάπτυξης του συστήματος.
- Φάση εξακρίβωσης αναγκών και Καθορισμού Απαιτήσεων. Κατά τη φάση εξακριβώθηκαν οι πραγματικές ανάγκες και καθορίστηκαν τα λειτουργικά χαρακτηριστικά ώστε να καλύπτονται οι ανάγκες.
- Φάση Καθορισμού Προδιαγραφών. Στη φάση αυτή καθορίσαμε επακριβώς το τι θα κάνει το σύστημα καθώς και τους περιορισμούς που θα έχει.
- Φάση Σχεδίασης. Με βάση τις προδιαγραφές που τέθηκαν προηγούμενος προχωρήσαμε στη σχεδίαση του συστήματος (οθόνες διεπαφής, αλγόριθμοι, αναφορές, βάση δεδομένων).
- Φάση Υλοποίησης. Σε αυτή τη φάση υλοποιήσαμε τα όσα σχεδιάστηκαν (σελίδες php, βάση δεδομένων MySql)
- Φάση Ελέγχου. Για να γίνει σωστός έλεγχος του συστήματος, δημιουργήθηκε ένας πίνακας με σενάρια ελέγχου. Επίσης το σύστημα δοκιμάστηκε από μια μικρή ομάδα καθηγητών και μαθητών.

Στα κεφάλαια που ακολουθούν πέρα από την περιγραφή των πιο πάνω φάσεων, γίνεται παρουσίαση και ανάλυση κάποιων αλγορίθμων για τους οποίους οι μαθητές θα κληθούν να λύσουν ασκήσεις.

ΚΕΦΑΛΑΙΟ 2

Προκαταρκτική Έρευνα

Η ανάπτυξη ενός συστήματος είναι μια σοβαρή και πολλές φορές ακριβή επένδυση, η οποία μπορεί να αποτύχει αν δεν ακολουθηθεί πιστά μια προγραμματισμένη διαδικασία. Η προκαταρκτική έρευνα έχει ως στόχο την διερεύνηση της αναγκαιότητας αλλά και της βιωσιμότητας ενός συστήματος. Μέσα από αυτή τη φάση εξετάσαμε τα ακόλουθα:

- Αναγνώριση του προβλήματος και εισήγηση λύσεων
- Καταγραφή των πλεονεκτημάτων που θα προκύψουν με τη δημιουργία του νέου συστήματος.
- Αναγνώριση των περιορισμών που θα έχει το σύστημα
- Αξιολόγηση της σκοπιμότητας της δημιουργίας του συστήματος
- Κοστολόγηση της ανάπτυξης αλλά και της συντήρησης του συστήματος.

2.1 Αναγνώριση Προβλήματος

Όπως αναφέραμε και στην εισαγωγή οι Κύπριοι μαθητές που προετοιμάζονται για την ολυμπιάδα πληροφορικής δεν έχουν στη διάθεση τους ένα σύστημα εκπαίδευσης που θα ανταποκρίνεται στις ανάγκες τους. Παρόλο που τα υπάρχοντα συστήματα είναι πολύ καλά, ξεκινούν από ένα επίπεδο πολύ πιο ψηλό από το επίπεδο των δικών μας μαθητών. Αν πάρουμε για παράδειγμα το Hellenico (το ελληνικό σύστημα), θα διαπιστώσουμε ότι έχει μόνο 4 εισαγωγικές ασκήσεις , γεγονός που δε βοηθά καθόλου τους άπειρους μαθητές.

Από τον πρώτο χρόνο ενασχόλησης μου με την παγκύπρια ολυμπιάδα πληροφορικής το 2009, μου ήταν ξεκάθαρο ότι για να προχωρήσουμε και να βελτιωθούμε θα έπρεπε να έχουμε το δικό μας εργαλείο εκπαίδευσης προσαρμοσμένο στο επίπεδο των δικό μας μαθητών. Σαν ενδιάμεση λύση χρησιμοποιήσαμε για πρώτη φορά το 2011, το CMS – Contest Management System, ένα σύστημα ανοικτού κώδικα που χρησιμοποιείται για τη διεξαγωγή διαγωνισμών πληροφορικής. Το CMS είναι ένα σύστημα εξαιρετικό για διαγωνισμούς (χρησιμοποιήθηκε σε πολλούς διεθνείς διαγωνισμούς όπως IOI 2012, IOI2013, BOI 2013), όμως έχει πολλές ελλείψεις όταν χρησιμοποιείται σαν σύστημα εκπαίδευσης.

Για την επίλυση του προβλήματος διαπίστωσα ότι υπήρχαν 3 λύσεις:

1. Αγορά ενός υφιστάμενου συστήματος (π.χ. Hellenico)

2. Εξέλιξη ενός υπάρχοντος συστήματος ανοικτού κώδικα

3. Δημιουργία ενός νέου συστήματος.

Η πρώτη περίπτωση διερευνήθηκε σοβαρά. Οι δημιουργοί του συστήματος μας έδωσαν κάποια δικαιώματα χρήσης ως administrators, οπότε μπορέσαμε να δούμε τα πλεονεκτήματα και τα μειονεκτήματα του Hellenico. Τα μειονεκτήματα του συστήματος τα οποία μας απέτρεψαν να προχωρήσουμε ήταν ότι το project είχε σταματήσει να αναπτύσσεται και ότι θα έπρεπε να γίνουν αρκετές αλλαγές για να το φέρουμε στα μέτρα μας. Η δεύτερη περίπτωση δεν μπορούσε να προχωρήσει αφού το CMS είναι δομημένο για διαγωνισμούς και όχι για εκπαίδευση. Η λύση που απέμενε ήταν η δημιουργία ενός συστήματος από εμάς για εμάς. Η προφανή δυσκολία ήταν ότι θα έπρεπε κάποιος να υλοποιήσει αυτό το σύστημα, κάτι που φιλοδοξεί να κάνει αυτή η διπλωματική εργασία.

2.2 Πλεονεκτήματα και Περιορισμοί

Τα πλεονεκτήματα που θεωρώ ότι θα προσφέρει η δημιουργία ενός νέου συστήματος είναι:

- Αυτονομία στους διοργανωτές της ολυμπιάδας πληροφορικής να οργανώσουν τις ασκήσεις με βάση τις ανάγκες των μαθητών μας

- Θα δώσει τη δυνατότητα στους μαθητές να έχουν πρόσβαση σε θεωρία που σχετίζεται με τις ασκήσεις

- Θα μπορεί να εμπλουτίζεται συνεχώς με καινούριες ασκήσεις.

- Το σύστημα θα έχει σχετικό πίνακα βαθμολογίας των χρηστών (Ranking), ώστε να υπάρχει ένα επιπλέον κίνητρο.

- Το σύστημα θα μπορεί να χρησιμοποιηθεί και από το Υπουργείο Παιδείας για τη διδασκαλία του μαθήματος της Γ Λυκείου.

Πολλά άλλα πλεονεκτήματα μπορούν να αναδειχθούν μετά την εφαρμογή του συστήματος.

Όσο αφορά τους περιορισμούς που θα έχει το σύστημα αυτοί συνοψίζονται στους πιο κάτω:

- Τεχνικοί περιορισμοί: Αριθμός χρηστών που θα μπορούν να υποβάλουν ταυτόχρονα στο σύστημα, αριθμός χρηστών που θα είναι ταυτόχρονα ενωμένοι στο σύστημα.

- Στο σύστημα θα μπορούν να υποβάλλονται μόνο ασκήσεις στις γλώσσες προγραμματισμού Pascal, C/C++.

- Τα αρχεία για εκφωνήσεις ασκήσεων, test cases, θεωρία θα πρέπει να υποβάλλονται στο σύστημα με συγκεκριμένη μορφή, που θα παρουσιαστεί πιο κάτω.

Κάποιοι από τους περιορισμούς θα μπορούσαν να εξαλειφτούν σε μελλοντικές αναβαθμίσεις του προγράμματος.

2.2 Μελέτη Σκοπιμότητας

Παρόλο που η σκοπιμότητα ενός έργου μπορεί να αλλάξει ανάλογα με τα δεδομένα που προκύπτουν σε κάθε φάση της ανάπτυξης, θεώρησα ότι στη δική μας περίπτωση τα δεδομένα ήταν ξεκάθαρα και οι αποκλίσεις από τους αρχικούς στόχους θα έπρεπε να είναι μικρές, οπότε η μελέτη σκοπιμότητας δεν αναθεωρήθηκε κατά τη διάρκεια υλοποίησης του συστήματος. Η αξιολόγηση της σκοπιμότητας του συστήματος έγινε με βάση 3 κατηγορίες όπως φαίνεται στο πιο κάτω σχήμα

Λειτουργική	Τεχνική	Οικονομική
• Αξίζει τον κόπο; • Θα το δεχτούν οι χρήστες • Επιλύει τα προβλήματα;	• Διαθέτουμε την τεχνολογία; • Υπάρχουν ικανοί χρήστες;	• Τι κόστος υπάρχει;

Ο παρακάτω πίνακας ανάλυσης σκοπιμότητας μας δείχνει την αξιολόγηση του συστήματος.

Αξιολόγηση Συστήματος	
Λειτουργική Σκοπιμότητα	Όπως έχει ήδη αναλυθεί το σύστημα αξίζει να δημιουργηθεί και οι χρήστες-μαθητές είναι έτοιμοι να το χρησιμοποιήσουν μιας και έχουν εμπειρία με παρόμοια συστήματος. Τέλος η δημιουργία του συστήματος επιλύει τα προβλήματα που υπάρχουν.
Τεχνική Σκοπιμότητα	Σίγουρα διαθέτουμε την τεχνολογία. Για την υλοποίηση του συστήματος θα χρησιμοποιήσουμε τις τεχνολογίες Apache – MySql – PHP, σε περιβάλλον windows 7. Οι δοκιμές θα γίνουν σε προσωπικό υπολογιστή, αλλά στη συνέχεια το πρόγραμμα θα πρέπει να εγκατασταθεί σε έναν κατάλληλο server. Όσο αφορά την ικανότητα των χρηστών, θα πρέπει να γίνει εκπαίδευση στους χρήστες-καθηγητές.
Οικονομική Σκοπιμότητα	Το κόστος του συστήματος αναλογεί στο κόστος που θα έχει ο server. Επίσης θα χρειαστεί να αγοραστεί ένα domain name και να φιλοξενηθεί ο server σε έναν κατάλληλο χώρο με πρόσβαση στο internet. Μια πρώτη εκτίμηση είναι ότι το σύστημα δεν θα υπερβεί το ποσό των 800 ευρώ

ΚΕΦΑΛΑΙΟ 3
ΠΡΟΔΙΑΓΡΑΦΗ ΑΠΑΙΤΗΣΕΩΝ

Σε αυτό το κεφάλαιο θα παρουσιάσω μια λεπτομερή περιγραφή του συστήματος. Για να το πετύχω αυτό θα χρησιμοποιήσω τόσο το γραπτό λόγο, αλλά και διαγράμματα ροής δεδομένων, οντοτήτων συσχετίσεων). Το κεφάλαιο αυτό χωρίζεται σε 3 υποκεφάλαια.

1. Γενική περιγραφή του συστήματος.

2. Απαιτήσεις των εξωτερικών παραγόντων.

3. Λειτουργικές απαιτήσεις.

Στη γενική περιγραφή του συστήματος θα περιγραφούν οι γενικές λειτουργίες του συστήματος, τα χαρακτηριστικά των χρηστών και οι περιορισμοί του συστήματος.

Στις απαιτήσεις των εξωτερικών παραγόντων θα περιγράφονται οι απαιτήσεις των χρηστών αλλά και οι απαιτήσεις σε υλικό και λογισμικό.

Τέλος στις λειτουργικές απαιτήσεις θα παρουσιάσω με λεπτομέρεια τις λειτουργίες του συστήματος, τις εισόδους και τις εξόδους καθώς και τη βάση δεδομένων του συστήματος.

Έχω καταλήξει στις παρακάτω προδιαγραφές μετά από μελέτη υπαρχόντων συστημάτων, συλλογή ερωτηματολογίων (Παράρτημα Α), συνεντεύξεις (Παράρτημα Β) και προσωπική εμπειρία.

3.1 Γενική Περιγραφή Συστήματος

Το σύστημα μας θα είναι online και για τη χρήση του θα απαιτείται να γίνεται εγγραφή. Θα υπάρχουν τρεις κατηγορίες χρηστών:

- Student. Σε αυτή την κατηγορία μπορεί να ενταχθεί ο οποιοσδήποτε φτάνει να κάνει εγγραφή στο σύστημα. Οι χρήστες αυτής της κατηγορίες θα μπορούν να υποβάλουν τις λύσεις των ασκήσεων, να λαμβάνουν την κατάλληλη ανατροφοδότηση και γενικά να χρησιμοποιούν το σύστημα χωρίς όμως περαιτέρω δικαιώματα.

- Teacher. Οι χρήστες αυτής της κατηγορίας θα μπορούν να δημιουργούν ασκήσεις και να τις υποβάλλουν στο σύστημα. Θα μπορούν επίσης να τροποποιούν και να διαγράφουν ασκήσεις τις οποίες έχουν υποβάλει οι ίδιοι. Τα δικαιώματα Teacher θα δίνονται από τον Administrator

- Administrator. Θα υπάρχει πολύ περιορισμένος αριθμός που θα ανήκει σε αυτή τη κατηγορία (2-3). Ο χρήστης με δικαιώματα Administrator έχει πλήρη έλεγχο του συστήματος. Αυτό σημαίνει ότι θα μπορεί να διαγράφει/τροποποιεί χρήστες, διαγράφει/τροποποιεί ασκήσεις και να αλλάζει τις ρυθμίσεις του συστήματος.

Οι δύο βασικές λειτουργίες του συστήματος θα είναι:

- Δημιουργία ασκήσεων και των αντιστοίχων test cases από τους Teachers/Administrators

- Υποβολή των λύσεων από τους μαθητές και έλεγχος των λύσεων από το σύστημα

Όσο αφορά τους περιορισμούς του συστήματος θα εξαρτώνται κατά κύριο λόγο από το υλικό που θα χρησιμοποιηθεί.

3.2 Απαιτήσεις των Εξωτερικών Παραγόντων

Οι απαιτήσεις των εξωτερικών παραγόντων περιγράφονται στον πιο κάτω πίνακα. Οι απαιτήσεις των χρηστών θα συμβολίζονται με το γράμμα U, του υλικού με το γράμμα H και του λογισμικού με το γράμμα S [02].

U1	Ο χρήστης θα κάνει login χρησιμοποιώντας το username και το password του
U2	Ο μελλοντικός χρήστης θα μπορεί να κάνει εγγραφή συμπληρώνοντας τη φόρμα

	εγγραφής
U3	Όταν κάνει login θα μπορεί να αλλάζει τα προσωπικά του στοιχεία (εκτός από το username)
U4	Θα έχει πρόσβαση μόνο σε ασκήσεις που δικαιούται με βάση τη βαθμολογία του.
U5	Θα βαθμολογείται με βάση τη δυσκολία της άσκησης
U6	Θα έχει πρόσβαση σε προηγούμενες υποβολές του
U7	Θα έχει πρόσβαση στη σύντομη θεωρεία του κεφαλαίου
U8	Θα κάνει υποβολή της λύσης του σε μια από τις γλώσσες Pascal ή C/C++
U9	Ο χρήστης teacher/administrator να μπορεί να ανεβάζει νέα άσκηση στο σύστημα
U10	Ο χρήστης teacher να μπορεί να τροποποιεί (update/delete) ασκήσεις που έχει ανεβάσει στο σύστημα.
U11	Ο χρήστης administrator να μπορεί να τροποποιεί (update/delete) ασκήσεις που έχει ανεβάσει στο σύστημα.
U12	Ο χρήστης administrator να μπορεί να διαγράψει, τροποποιήσει το λογαριασμό ενός χρήστη
H1	Ο υπολογιστής στον οποίο θα υλοποιηθεί το σύστημα θα πρέπει να έχει δυνατότητες εξυπηρέτησης πολλών ατόμων ταυτόχρονα. Αυτό συνεπάγεται με υψηλές απαιτήσεις σε πόρους τόσο της κεντρικής Μονάδας επεξεργασίας (CPU) όσο και της Κεντρικής Μνήμης (RAM).
H2	Θα πρέπει να υπάρχει εφεδρικός δίσκος για να γίνεται Back up το σύστημα.
H3	Πρέπει να υπάρχει κατάλληλη σύνδεση στο διαδίκτυο υψηλής ταχύτητας.

H4	Για λόγους Ασφαλείας συνιστάται η χρήση επιπλέον υπολογιστή που θα έχει το ρόλο του Firewall.
S1	Το λειτουργικό σύστημα της μηχανής μπορεί να είναι είτε Windows είτε Linux
S2	Ο server που θα χρησιμοποιηθεί θα είναι ο Apache
S3	Η βάση δεδομένων του συστήματος θα είναι η MySQL
S4	Για την κατασκευή και τον προγραμματισμό των ιστοσελίδων θα χρησιμοποιηθεί η PHP

Απαίτηση Χρήστη U1.

Εισάγοντας τη διεύθυνση του συστήματος θα εμφανίζεται στο χρήστη η αρχική σελίδα στην οποία θα μπορεί να κάνει login. Είσοδος: Το username και το password του χρήστη. Επεξεργασία: Επαλήθευση των στοιχείων με βάση του τι υπάρχει στη βάση δεδομένων. Έξοδος: Εμφάνιση της επόμενης σελίδας του συστήματος ή εμφάνιση μηνύματος λάθους.

Απαίτηση Χρήστη U2.

Στην αρχική σελίδα θα υπάρχει δυνατότητα εγγραφής νέου χρήστη. Για να γίνει αυτό θα πρέπει να συμπληρώνεται η σχετική φόρμα Είσοδος: Τα στοιχεία του νέου χρήστη Επεξεργασία: Τα στοιχεία του νέου χρήστη θα επικυρώνονται και στη συνέχεια θα καταχωρούνται στη βάση δεδομένων. Έξοδος: Ο χρήστης ενημερώνεται για την επιτυχία ή την αποτυχία της εγγραφής του.

Απαίτηση Χρήστη U3.

Αλλαγή ή ενημέρωση προσωπικών στοιχείων Είσοδος: Ενημερωμένα στοιχεία χρήστη Επεξεργασία: Τα στοιχεία επικυρώνονται και ενημερώνεται η βάση δεδομένων Έξοδος: Ο χρήστης ενημερώνεται για την επιτυχία ή την αποτυχία της αλλαγής στοιχείων του.

Απαίτηση Χρήστη U4.

Κάθε νέος χρήστης θα έχει πρόσβαση μόνο σε ένα περιορισμένο αριθμό ασκήσεων για να αποκτήσει πρόσβαση σε περεταίρω ασκήσεις θα πρέπει να συγκεντρώσει συγκεκριμένο αριθμό βαθμών Είσοδος: Το username και το password του χρήστη. Επεξεργασία: Γίνεται ανάκτηση της βαθμολογίας του χρήστη και αναλόγως εμφανίζονται οι επιτρεπόμενες ασκήσεις . Έξοδος: Εμφανίζονται οι ασκήσεις τις οποίες επιτρέπεται να έχει πρόσβαση ο χρήστης.

Απαίτηση Χρήστη U5.

Κάθε άσκηση του συστήματος θα ανήκει σε μια κατηγορία ανάλογα με τη δυσκολία της της (A, B, C) . Οι ασκήσεις της κατηγορίας Α θα βαθμολογείται με άριστα το 20, της Β με άριστα το 50 και της C με άριστα το 100. Είσοδος: Επιλογή άσκησης και υποβολή λύσης. Επεξεργασία: Βαθμολόγηση της λύσης με βάση τη δυσκολία της. Έξοδος: Η βαθμολογία που πέτυχε ο χρήστης.

Απαίτηση Χρήστη U6.

Ο χρήστης της θα μπορεί να βλέπει τα στοιχεία (πηγαίος κώδικας, αποτέλεσμα) των προηγούμενων του υποβολών στο σύστημα. Είσοδος: Επιλογή άσκησης για την οποία ο χρήστης έχει κάνει υποβολές. Επεξεργασία: Εύρεση των στοιχείων της υποβολής, που επέλεξε ο χρήστης. Έξοδος: Τα στοιχεία της υποβολής.

Απαίτηση Χρήστη U7.

Κάθε χρήστης θα έχει πρόσβαση στη θεωρία του αντίστοιχου κεφαλαίου ασκήσεων. Είσοδος: Επιλογή κεφαλαίου. Επεξεργασία: Εύρεση και παρουσίαση της θεωρίας που έχει επιλεγεί. Έξοδος: Τα αρχεία της θεωρίας (pdf, pptx, videos) για τη θεωρία που έχει επιλεγεί.

Απαίτηση Χρήστη U8.

Ο χρήστης θα επιλέγει να τη γλώσσα προγραμματισμού (Pascal, C/C++) στην οποία θα υποβάλει τη λύση του. Είσοδος: Επιλογή της γλώσσας και υποβολή αντίστοιχου κώδικα. Επεξεργασία: Αποσφαλμάτωση του πηγαίου κώδικα, έλεγχος των test cases της άσκησης και βαθμολόγηση. Έξοδος: Αποτέλεσμα αξιολόγησης του κώδικα που έχει υποβληθεί.

Απαίτηση Χρήστη U9.

Ο χρήστης teacher/administrator θα μπορεί να ετοιμάζει και να δημιουργεί στο σύστημα νέες ασκήσεις. Είσοδος: Για κάθε άσκηση ο teacher/administrator θα πρέπει να υποβάλλει στο σύστημα την εκφώνηση και τα test cases. Επίσης θα πρέπει να επιλέγει την κατηγορία της άσκησης καθώς και το βαθμό δυσκολίας της. Επεξεργασία: Αποθήκευση της άσκησης στο σύστημα. Έξοδος: Παρουσίαση της άσκησης στους άλλους χρήστες.

Απαίτηση Χρήστη U10.

Ο χρήστης teacher θα μπορεί να τροποποιεί ασκήσεις που έχει υποβάλει στο σύστημα. Είσοδος: Ο teacher θα επιλέγει από την λίστα των ασκήσεων, που ο ίδιος έχει ανεβάσει, αυτή που θέλει να τροποποιήσει. Στη συνέχεια θα επιλέγει αν θέλει να διαγράψει ή αν θέλει να τροποποιήσει την άσκηση. Επεξεργασία: Η άσκηση διαγράφεται ή τροποποιείται. Έξοδος: Η άσκηση διαγράφεται από το σύστημα ή εμφανίζεται τροποποιημένη.

Απαίτηση Χρήστη U11.

Ο χρήστης administrator θα μπορεί να τροποποιήσει ασκήσεις, όπως ο teacher, με τη διαφορά ότι θα μπορεί να το κάνει για οποιαδήποτε άσκηση και όχι μόνο αυτές που έχει ανεβάσει ο ίδιος.

Απαίτηση Χρήστη U12.

Ο administrator θα έχει πρόσβαση στα στοιχείων όλων των χρηστών. Θα μπορεί να διαγράψει κάποιον χρήστη ή να τροποποιήσει τα στοιχεία του. Είσοδος: Επιλογή χρήστη. Επιλογή για delete ή update. Επεξεργασία: Ο χρήστης διαγράφεται ή τα στοιχεία του τροποποιούνται. Έξοδος: Ο χρήστης δεν εμφανίζεται πλέον στο σύστημα ή εμφανίζεται με αλλαγμένα στοιχεία.

Απαίτηση Υλικού H1

Ο υπολογιστής στον οποίο θα υλοποιηθεί το σύστημα θα πρέπει να έχει δυνατότητες εξυπηρέτησης πολλών ατόμων ταυτόχρονα. Αυτό συνεπάγεται με υψηλές απαιτήσεις σε πόρους τόσο της κεντρικής Μονάδας επεξεργασίας (CPU) όσο και της Κεντρικής Μνήμης (RAM). Η επιλογή του υπολογιστή που θα επιλεγεί εξαρτάται και από το ποσό που είναι διαθέσιμος να ξοδέψει ο οργανισμός που θα χρησιμοποιήσει το σύστημα. Στον παρακάτω πίνακα προτείνω 2 λύσεις. Η μία χρησιμοποιεί έναν επιτραπέζιο υπολογιστή σαν server και η άλλη ένα σύστημα NAS.

Πρόταση 1	Πρόταση 2
Επεξεργαστής INTEL CORE I3-3225 3.30GHZ LGA1155 - BOX **Μνήμη RAM** KINGSTON KHX16C9B1RK2/8 8GB (2X4GB) DDR3 1600MHZ HYPERX RED SERIES DUAL CHANNEL KIT **Mother Board** ASUS P8Z77-V LX2 RETAIL **Σκληρός Δίσκος** 2X TOSHIBA DT01ACA050 500GB SATA3 **Τιμή: 600 ευρώ**	**Synology DS413** **Επεξεργαστής** Dual Core 1.067 GHz **Μνήμη RAM** 1 GB DDR3 **Σκληρός Δίσκος** 2X TOSHIBA DT01ACA050 500GB SATA3 **Τιμή: 520 ευρώ**

Απαίτηση Υλικού Η2

Θα πρέπει να υπάρχει εφεδρικός δίσκος για να γίνεται Back up το σύστημα. Και στις 2 προτάσεις που έχουν αναφερθεί πιο πάνω υπάρχουν 2 σκληροί δίσκοι. Ο ένας δίσκος θα είναι ο κύριος (main), όπου θα είναι εγκατεστημένο το σύστημα, ενώ ο δεύτερος θα χρησιμοποιείται ως εφεδρικός. Αν επιλεγεί το σύστημα της πρότασης 1 του πιο πάνω πίνακα τότε θα πρέπει να γίνει εγκατάσταση κατάλληλου λογισμικού που θα κάνει αυτόματα backup τον «κύριο» δίσκο στον εφεδρικό σε τακτά χρονικά διαστήματα.. Στην περίπτωση του NAS συστήματος αυτό γίνεται πιο εύκολα αφού υπάρχει έτοιμη εργασία ειδικά για αυτό τον σκοπό, εγκατεστημένη στο σύστημα.

Απαίτηση Υλικού Η3

Η Παγκύπρια Ολυμπιάδα Πληροφορικής έχει περίπου 100-120 συμμετέχοντες κάθε χρόνο. Το σύστημα πρέπει να καλύπτει τουλάχιστον αυτούς. Μια γρήγορη γραμμή Internet είναι απαραίτητη για την ομαλή λειτουργία του συστήματος. Θεωρώ ότι πρέπει να υπάρχει μια γραμμή με ελάχιστη ταχύτητα 4Mb/s.

Απαίτηση Υλικού Η4

Για λόγους Ασφαλείας συνιστάται η χρήση επιπλέον υπολογιστή που θα έχει το ρόλο του Firewall. Η 2η πρόταση που προτάθηκε έχει εγκατεστημένα πολλά χαρακτηρίστηκα ασφάλειας (π.χ. firewall) που απλά πρέπει να ενεργοποιηθούν.

Απαίτηση Λογισμικού S1

Το λειτουργικό σύστημα που θα χρησιμοποιήσω είναι τα Windows 7. Με μικρές τροποποιήσεις το σύστημα μπορεί να τρέξει και σε λειτουργικό Linux

Απαίτηση Λογισμικού S2

Σαν server θα χρησιμοποιηθεί ο Apache που είναι μια δωρεάν και αξιόπιστη λύση.

Απαίτηση Λογισμικού S3

Σαν βάση δεδομένων θα χρησιμοποιηθεί η MySql, για τους ίδιους λόγους που αναφέρθηκαν πιο πάνω.

Απαίτηση Λογισμικού S4

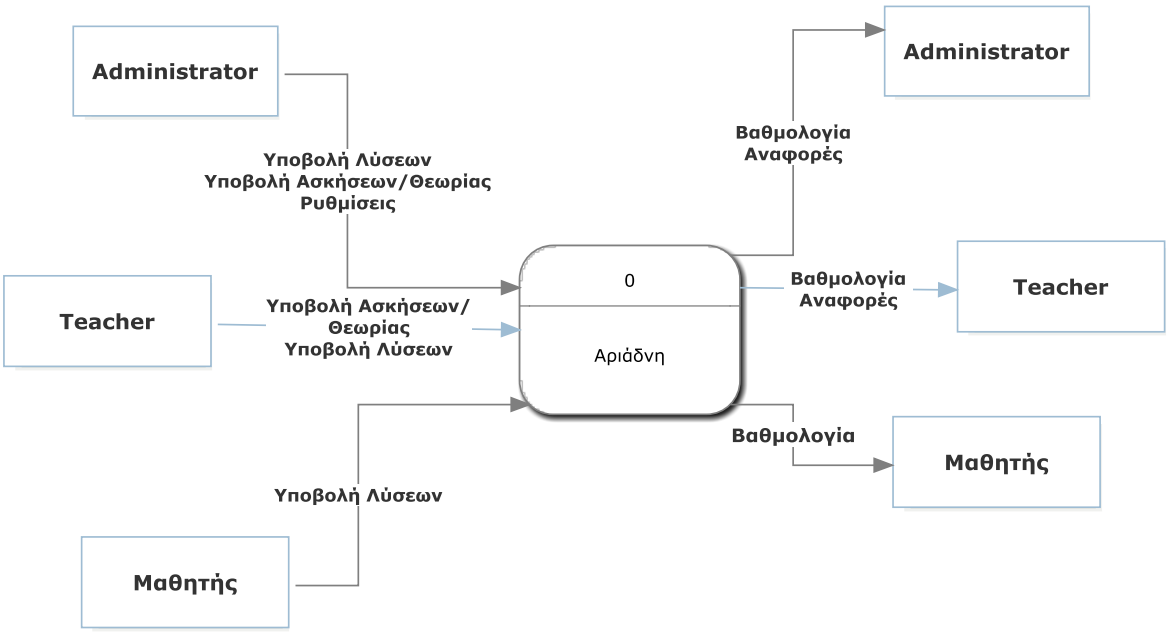
Το σύστημα θα είναι ένα web application και για την υλοποίηση του θα χρησιμοποιήσω την PHP.

3.3 Λειτουργικές Απαιτήσεις

Για να δείξω τις λειτουργικές απαιτήσεις του συστήματος αποφάσισα να χρησιμοποιήσω τα διαγράμματα ροής δεδομένων (ΔΡΔ) τα οποία θα παρουσιάζουν τη ροή των δεδομένων μεταξύ του εξωτερικού περιβάλλοντος του συστήματος (χρήστες) και του εσωτερικού περιβάλλοντος (διαδικασίες, βάσης δεδομένων). Τα ΔΡΔ παριστάνονται σε διαφορετικά επίπεδα λεπτομέρειας. Στο πρώτο, λιγότερο λεπτομερές επίπεδο, παριστάνεται ολόκληρο το σύστημα ως ένας μετασχηματισμός σύνθετων δεδομένων. Ο μετασχηματισμός αυτός αναλύεται σε περισσότερα επίπεδα λεπτομέρειας, μέχρι το σημείο που ο θεώρησα ότι γίνεται ικανοποιητική περιγραφή. Στα ΔΡΔ θα χρησιμοποιήσω το συμβολισμό που προτείνεται από τους Yourdon/DeMarco [01][02].

3.3.1 Διαγράμματα Ροής Δεδομένων

Στο σχήμα 1 φαίνεται το γενικό διάγραμμα του συστήματος. Το σύστημα θα ονομάζεται «**ΑΡΙΑΔΝΗ»** και με τη βοήθεια της θα προσπαθήσουμε να βγούμε από το λαβύρινθο των προβλημάτων.



Σχήμα 1: Το ΔΡΔ επιπέδου 0 του συστήματος.

Στο ΔΡΔ επιπέδου 0 φαίνεται ότι το σύστημα θα περιλαμβάνει τρεις κατηγορίες χρηστών (Administrator, Teacher, Μαθητής). Κάποιες λειτουργίες θα είναι κοινές και για τους τρεις ενώ κάποιες άλλες όχι.

Το σύστημα θα είναι διαδραστικό, δηλαδή θα υπάρχει αμφίδρομη επικοινωνία μεταξύ χρήστη και συστήματος.

Στο σχήμα 2 βλέπουμε την πρώτη αποσύνθεση του συστήματος σε διαδικασίες για το χρήστη μαθητή, στο σχήμα 3 για το χρήστη teacher και στο σχήμα 4 για το χρήστη administrator.

Οι βασικές διαδικασίες του συστήματος είναι:

1. **SignUp**. Εγγραφή στο σύστημα

2. **Login**. Πρόκειται για τη λειτουργία εισόδου στο σύστημα.

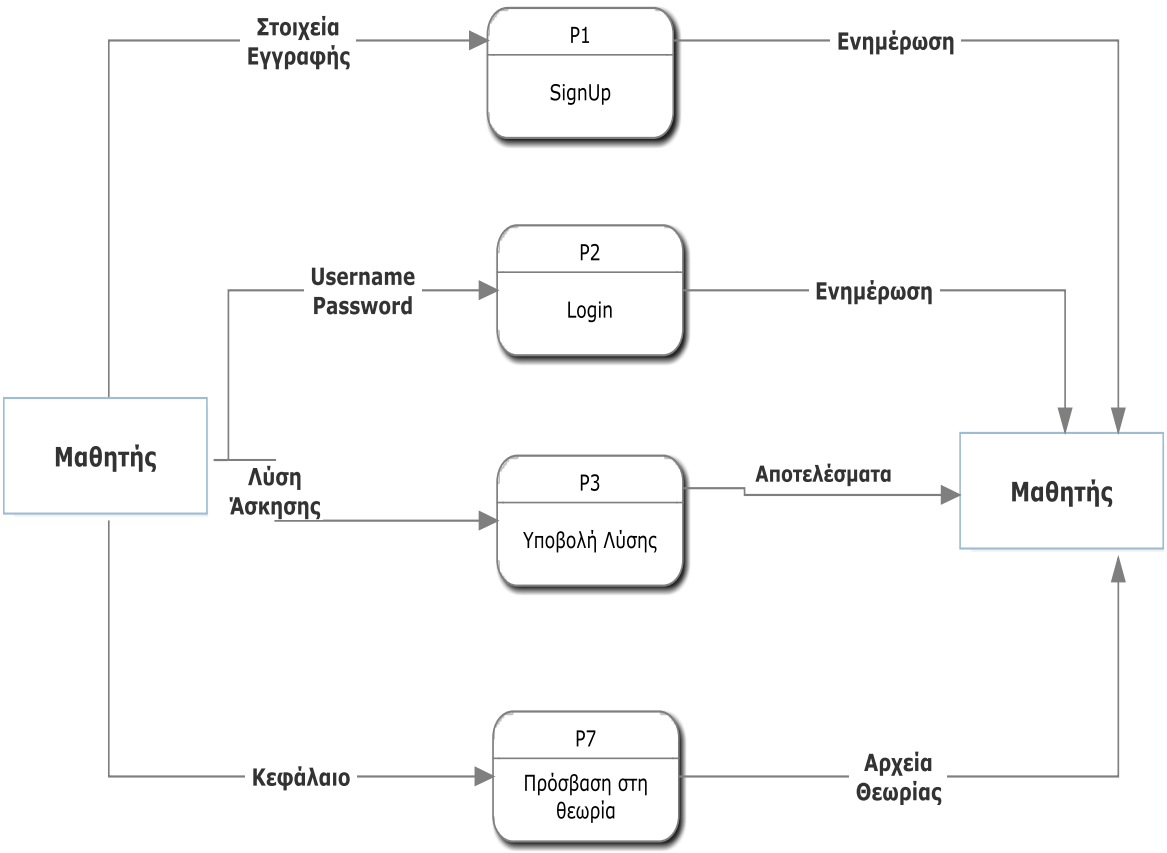
3. **Υποβολή Λύσης**. Πρόκειται για τη βασικότερη λειτουργία του συστήματος. Ο χρήστης θα υποβάλει τη λύση του για μία άσκηση και η λύση αυτή θα αξιολογείται από το σύστημα.

4. **Υποβολή/Διόρθωση/Διαγραφή άσκησης**. Λειτουργία μόνο για τους teachers και admiistrators ώστε να μπορούν να αναρτούν, να διορθώνουν ή να διαγράφουν ασκήσεις.

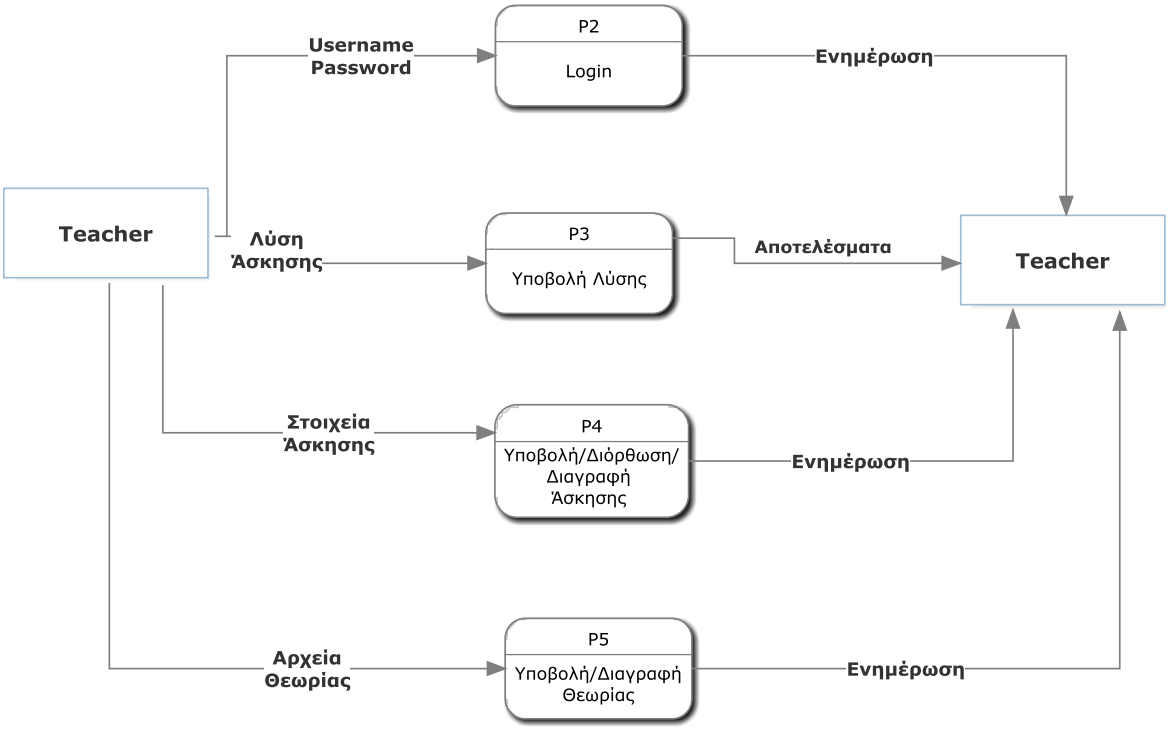
5. **Υποβολή/ Διαγραφή Θεωρίας**. Λειτουργία μόνο για τους teachers και admiistrators ώστε να μπορούν να αναρτούν ή να διαγράφουν αρχεία θεωρίας.

6. **Αλλαγή Ρυθμίσεων**. Λειτουργία μόνο για τον administrators, ώστε να αλλάζει ο τύπος και τα υπόλοιπα στοιχεία των χρηστών.

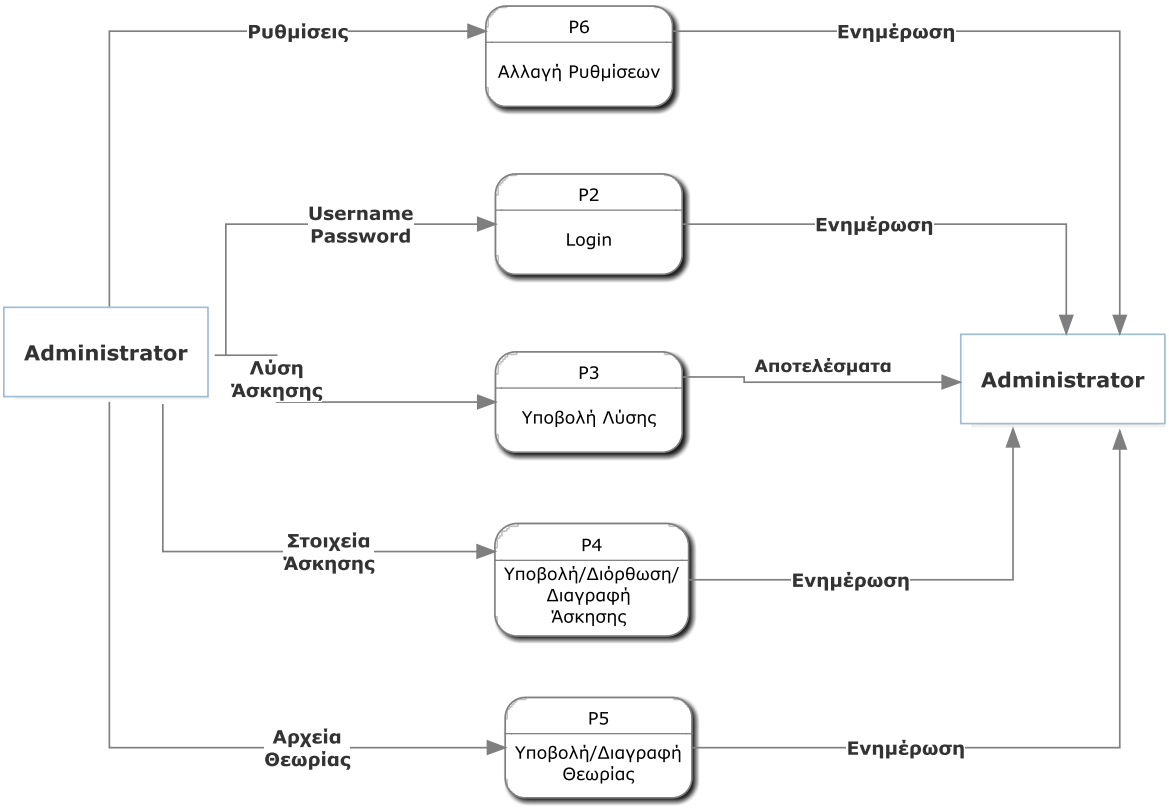
7. **Πρόσβαση στη θεωρία.** Μέσα από αυτή τη λειτουργία ο χρήστης θα έχει πρόσβαση στη θεωρία του κάθε κεφαλαίου.



Σχήμα 2: Το ΔΡΔ επιπέδου 1 για το χρήστη μαθητή.



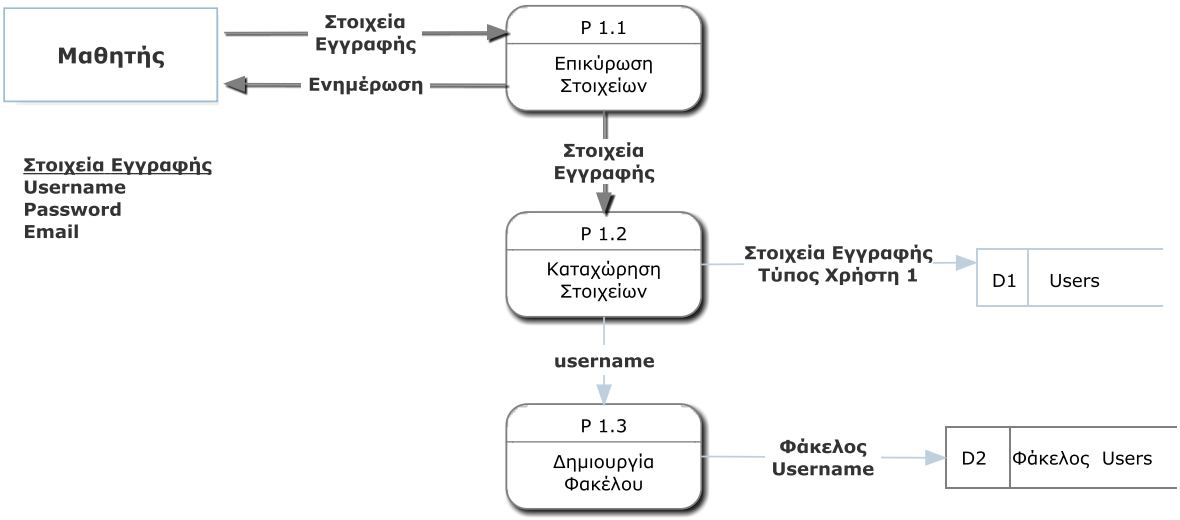
Σχήμα 3: Το ΔΡΔ επιπέδου 1 για το χρήστη teacher.



Σχήμα 4: Το ΔΡΔ επιπέδου 1 για το χρήστη Administrator.

P.1 Sign Up

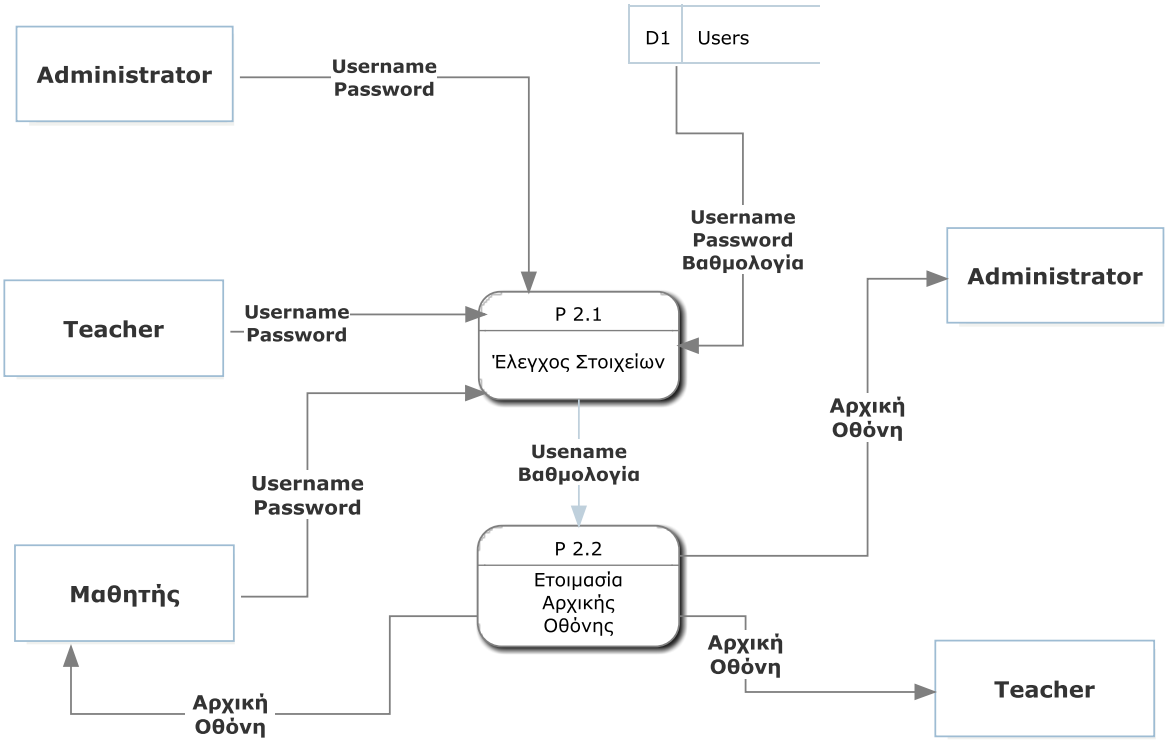
Η διαδικασία της εγγραφής (signup) παρουσιάζεται στο σχήμα 5. Ο χρήστης administrator δημιουργείται αυτόματα με το σύστημα. Όλοι οι υπόλοιποι χρήστες ξεκινούν με τα προνόμια του μαθητή. Ο νέος χρήστης δίνει στο σύστημα τα στοιχεία του (username, password, email,), το σύστημα επικυρώνει με τους κατάλληλους ελέγχους τα στοιχεία και στη συνέχεια τα καταχωρεί στο αρχείο χρηστών. Επίσης για κάθε νέο χρήστη δημιουργείται ένας φάκελος μέσα στο φάκελο users με το username του νέου χρήστη.



Σχήμα 5: Η διαδικασία εγγραφής (Sign Up)

P..2 Login

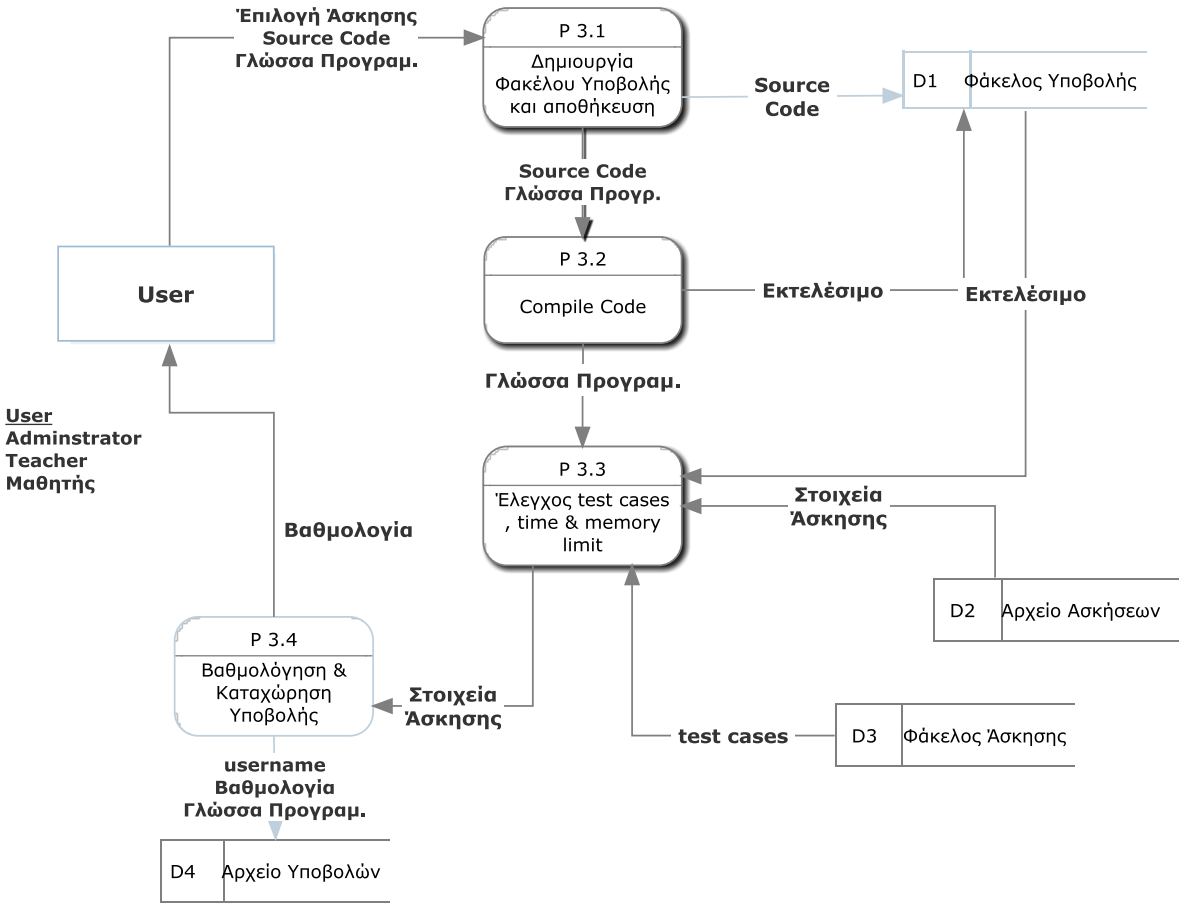
Στο σχήμα 6 βλέπουμε την ανάλυση της διαδικασίας Login. Ο χρήστης δίνει το username και password στο σύστημα, το οποίο ελέγχει την ορθότητα τους από το αρχείο users. Αν τα στοιχεία είναι σωστά τότε το σύστημα προχωρά στην ανάκτηση της βαθμολογίας και ετοιμάζει την αρχική οθόνη.



Σχήμα 6: Το ΔΡΔ για τη διαδικασία Login

P.3 Υποβολή λύσης

Η διαδικασία αυτή είναι κοινή και για τους τρεις τύπους χρήστη. Η διαδικασία αυτή ξεκινά με την επιλογή της άσκησης και την υποβολή από το χρήστη του κώδικα προς. Ταυτόχρονα πρέπει να επιλεγεί η γλώσσα προγραμματισμού (Pascal, C/C++). Με την υποβολή της λύσης δημιουργείται ένας φάκελος για να αποθηκευτούν τα στοιχεία της υποβολής. Στη συνέχεια ο κώδικας ελέγχεται για συντακτικά λάθη και δημιουργείται το εκτελέσιμο (executable) το οποίο αποθηκεύεται, όπως και ο πηγαίος κώδικας, στο φάκελο υποβολής. Ακολουθεί ο έλεγχος για την ορθότητα των test cases καθώς και για το αν το πρόγραμμα δεν ξεπερνά τα όρια μνήμης και χρόνου. Τα test cases βρίσκονται μέσα στο φάκελο της άσκησης. Μετά τους ελέγχους η άσκηση βαθμολογείται και τα στοιχεία της υποβολής καταχωρούνται στο αρχείο υποβολών. Τα αποτελέσματα παρουσιάζονται στο χρήστη. Στο σχήμα 7 φαίνεται το ΔΡΔ για την υποβολή άσκησης για αξιολόγηση.



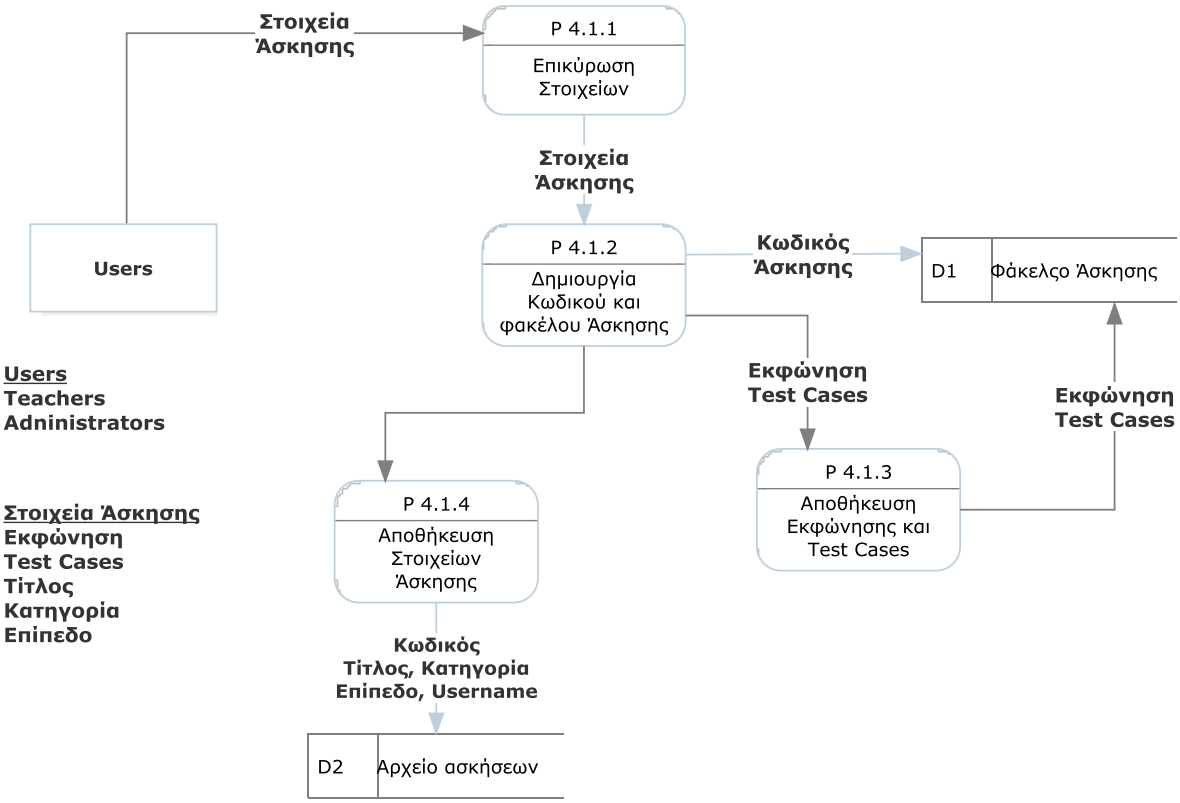
Σχήμα 7: Υποβολή Λύσης για αξιολόγηση

P.4 Υποβολή/ Διόρθωση/ Διαγραφή Άσκησης

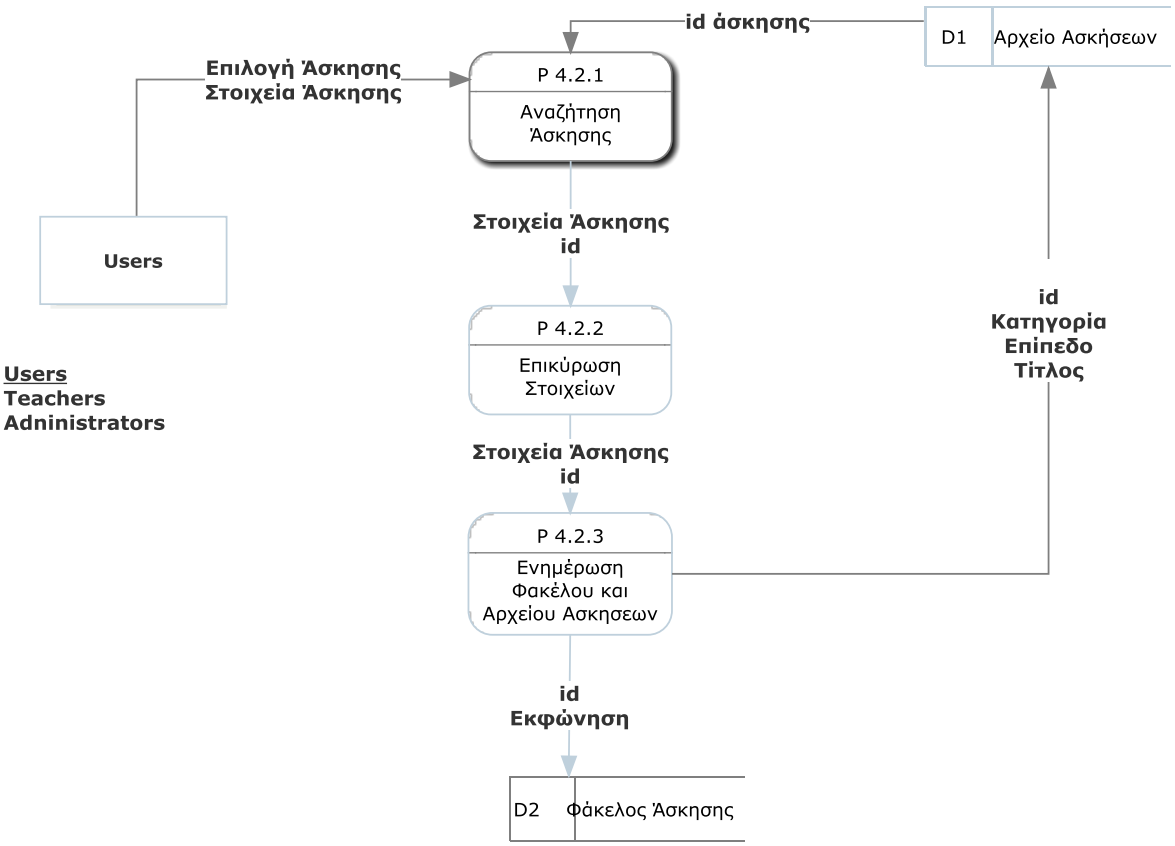
Οι χρήστες teacher και administrators έχουν τη δυνατότητα να αναρτούν ασκήσεις στο σύστημα. Θα μπορούν επίσης να τις διορθώνουν και να τις διαγράφουν. Στο σχήμα 8 παρουσιάζεται η διαδικασία ανάρτησης της άσκησης. Οι χρήστες υποβάλουν στο σύστημα τα στοιχεία της άσκησης (εκφώνηση, test cases, τίτλος, κατηγορία, επίπεδο), τα οποία αξιολογούνται για την ορθότητα τους. Αν τα στοιχεία είναι ορθά τότε δημιουργείται ένας φάκελος για την άσκηση με όνομα τον κωδικό της. Μέσα σε αυτόν το φάκελο αποθηκεύονται τα test cases και η εκφώνηση της άσκησης. Τέλος τα υπόλοιπα στοιχεία αποθηκεύονται στο αρχείο ασκήσεων Η διαδικασία παρουσιάζεται στο σχήμα 8.

Στην περίπτωση που ο χρήστης θέλει να διορθώσει την άσκηση υποβάλει τα στοιχεία που θέλει να διορθώσει. Τα στοιχεία επικυρώνονται και τότε ενημερώνονται ο φάκελος της άσκησης και το αρχείο ασκήσεων. Η διαδικασία παρουσιάζεται στο σχήμα 9.

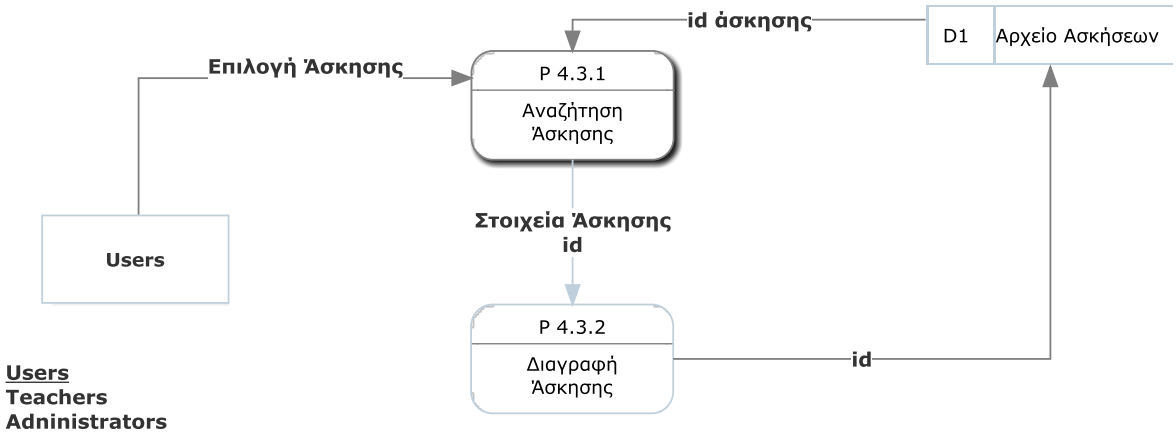
Τέλος στο σχήμα 10 παρουσιάζεται η διαδικασία διαγραφής μια άσκησης.



Σχήμα 8: ΔΡΔ Υποβολής άσκησης



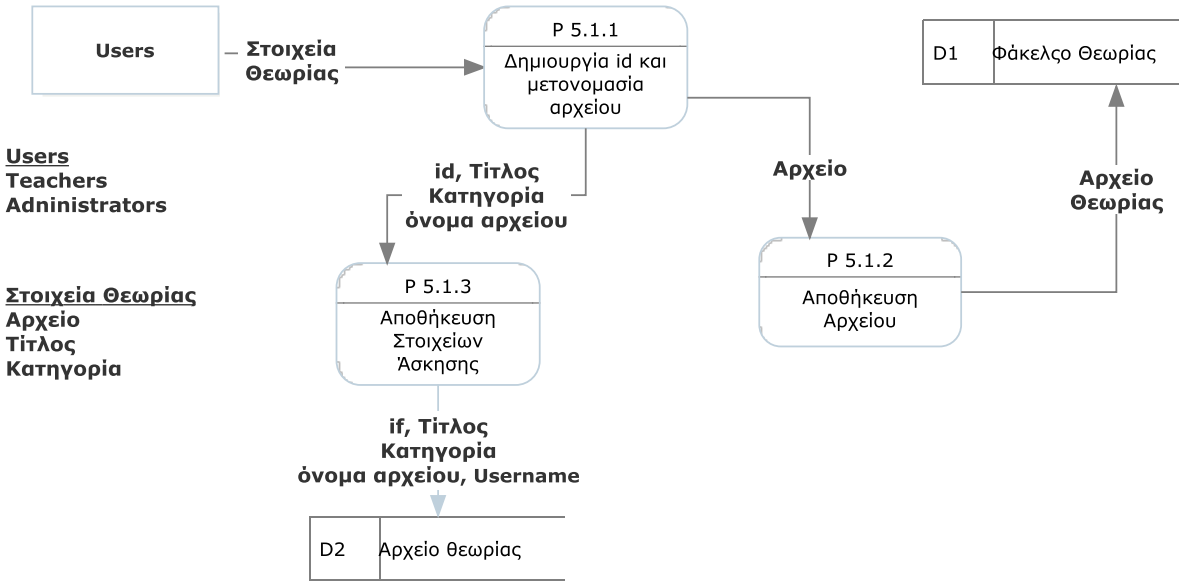
Σχήμα 9: ΔΡΔ Διόρθωση άσκησης



Σχήμα 10: Διαγραφή άσκησης

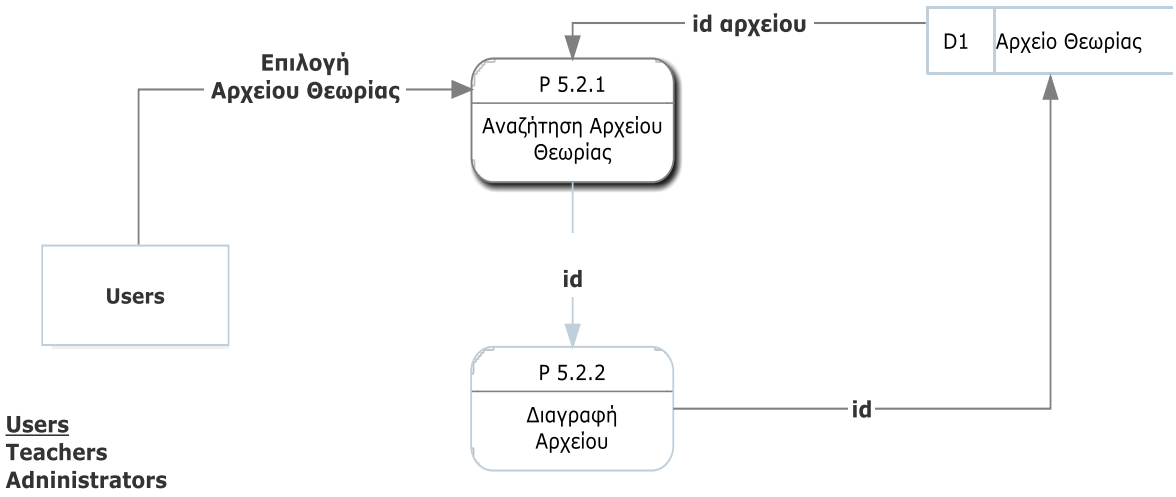
P.5 Υποβολή/ Διαγραφή Θεωρίας

Οι χρήστες teacher και administrators έχουν τη δυνατότητα να αναρτούν αρχεία θεωρίας στο σύστημα (Σχήμα 11). Θα έχουν επίσης τη δυνατότητα να τα διαγράφουν (Σχήμα 12).



Σχήμα 11: Υποβολή αρχείων θεωρίας.

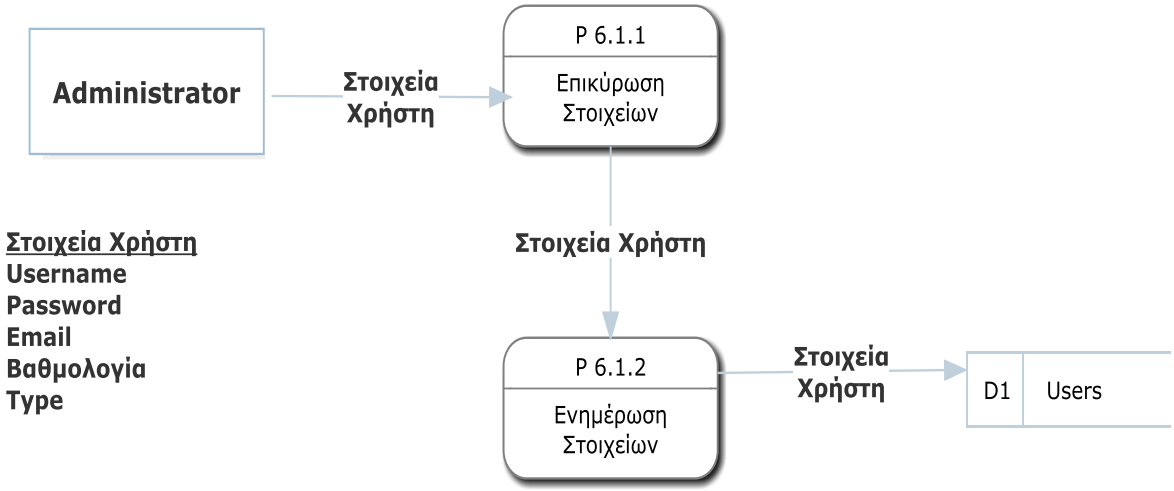
Ο χρήστης θα δίνει τα στοιχεία τους αρχείο που θέλει να αναρτήσει (τίτλος, κατηγορία). Όταν το αρχείο γίνεται upload, θα μετονομάζεται και θα αποθηκεύεται στο φάκελο θεωρίας. Τα υπόλοιπα στοιχεία θα αποθηκεύονται στο αρχείο θεωρίας.



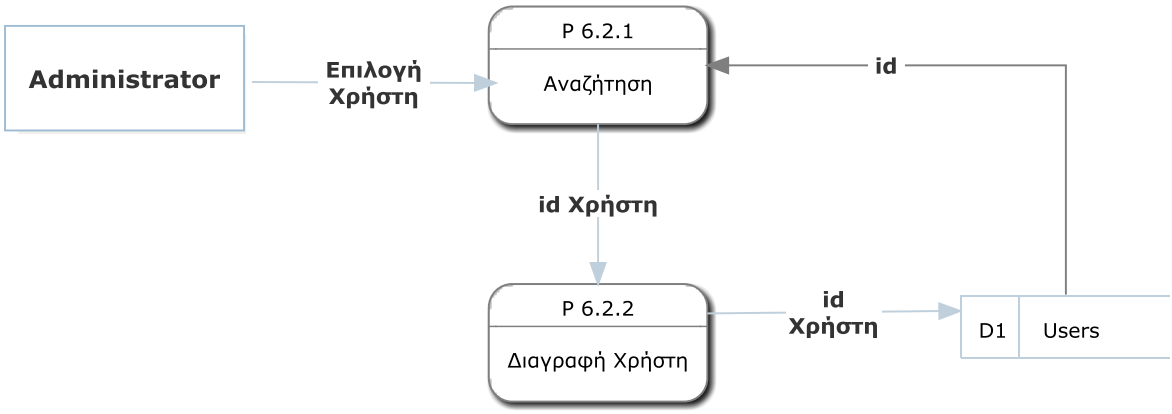
Σχήμα 12: Διαγραφή αρχείων θεωρίας.

P.6 Αλλαγή Ρυθμίσεων Χρήστη

Σε αυτή τη λειτουργία, πρόσβαση θα έχει μόνο ο administrator. Θα μπορεί να διορθώνει τα στοιχεία των χρηστών (π.χ. αλλαγή τύπου χρήστη) αλλά και να διαγράφει κάποιον χρήστη.



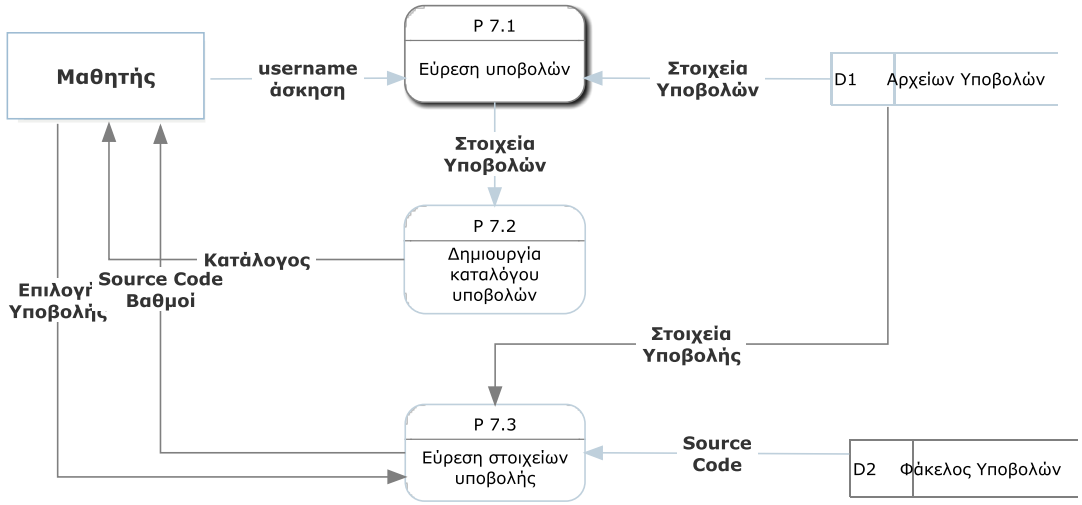
Σχήμα 13: Ενημέρωση στοιχείων χρήστη



Σχήμα 14: Διαγραφή Χρήστη

Ρ.7 Πρόσβαση σε παλαιότερες υποβολές

Ο χρήστης θα έχει πρόσβαση σε παλαιότερες υποβολές. Κάνοντας login θα επιλέγει την άσκηση και το σύστημα θα δημιουργεί έναν κατάλογο με τις υποβολές που έχουν γίνει. Ο χρήστης θα επιλέγει την υποβολή και το σύστημα θα του παρουσιάζει τα στοιχεία της συγκεκριμένης υποβολής (κώδικας, αποτελέσματα). Η διαδικασία παρουσιάζεται στο σχήμα 15.



Σχήμα 15: Πρόσβαση σε παλαιότερες υποβολές

Κεφάλαιο 4
Σχεδίαση Συστήματος

Στο κεφάλαιο γίνεται παρουσίαση της σχεδίασης του συστήματος. Παρουσιάζεται ο σχεδιασμός της βάσης δεδομένων (schema), οι διεπαφές (interface) του συστήματος με το χρήστη και ο τρόπος αποθήκευσης αρχείων στην πλευρά του server.

4.1 Σχεδιασμός Βάσης Δεδομένων

Όπως έχει αναφερθεί και πιο πάνω, η βάση δεδομένων θα δημιουργηθεί με τη χρήση της Mysql. Παρακάτω φαίνεται η ανάλυση για τον κάθε πίνακα.

Πίνακας: Users

Sql Statement

```
CREATE TABLE users(
    username varchar(20) NOT NULL,
    password varchar(20) NOT NULL,
    email varchar(30) NOT NULL,
    user_type int(1) NOT NULL,
```

```
    points int(11)
    PRIMARY KEY ( username ),
);
```

Ο πίνακας users περιέχει τα στοιχεία των χρηστών. Τα πεδία username, password, email δίνονται από τον χρήστη και είναι απαραίτητα. Το πεδίο user_type παίρνει τις τιμές 1 για απλό χρήστη, 2 για teacher, 3 για administrator. Το πεδίο αυτό μπορεί να αλλάξει μόνο από τον administrator. Όταν γίνεται εγγραφή από έναν χρήστη αυτό το πεδίο είναι εξ ορισμού 1. Το πεδίο points περιέχει τους βαθμούς του κάθε χρήστη. Πρωτεύον κλειδί (Primary Key) του πίνακα είναι το πεδίο username. Όλα τα στοιχεία του πίνακα είναι απαραίτητα (NOT NULL). Το username πρέπει να είναι μεγαλύτερο από 0 χαρακτήρες και μικρότερο από 21. Το password πρέπει να έχει τουλάχιστον 8 χαρακτήρες αλλά όχι περισσότερους από 20. Αυτοί οι έλεγχοι θα γίνονται από το σύστημα κατά τη διάρκεια της εγγραφής του νέου χρήστη.

Πίνακας: tb_problems

```
Sql Statement
CREATE TABLE tb_problems(
    prob_id int(11) NOT NULL,
    title varchar(255) NOT NULL,
    cat varchar(35) NOT NULL,
    level varchar(5) NOT NULL,
    user varchar(20) NOT NULL
    PRIMARY KEY (prob_id),
);
```

Ο πίνακας tb_problems δημιουργήθηκε για να αποθηκεύει τις πληροφορίες των προβλημάτων/ασκήσεων. Για κάθε πρόβλημα είναι απαραίτητο να έχουμε το prob_id (primary key), τον τίτλο (title), την κατηγορία στην οποία ανήκει (cat), το επίπεδο δυσκολίας του (level) και το συγγραφέα του προβλήματος (user). Είναι φανερό ότι από τον πίνακα λείπει η εκφώνηση και τα test cases.

Για λόγους που εγώ θεωρώ ότι προσφέρουν καλύτερη διαχείριση στο σύστημα η εκφώνηση και τα test cases αποθηκεύονται σε ένα φάκελο στον server. Όταν ένας χρήστης (teacher/ administrator) δημιουργήσει μια νέα άσκηση μέσα στο φάκελο **askiseis** δημιουργείται ένας φάκελος με όνομα το prob_id της άσκησης. Μέσα σε αυτόν το φάκελο αποθηκεύονται η εκφώνηση και τα test cases. Τόσο η εκφώνηση, όσο και τα test cases είναι σε μορφή plain text (txt)

Τόσο η κατηγορία (cat) όσο και επίπεδο(level) της άσκησης θα εισάγονται μέσα από την επιλογή ενός πτυσσόμενου μενού (combo box), οπότε δε χρειάζεται να προβούμε σε κάποιο έλεγχο για την ορθότητα τους.

Όλα τα πεδία είναι αναγκαία και πρωτεύων κλειδί είναι το prob_id.

Πίνακας: tb_submisions

Sql Statement

```
CREATE TABLE tb_submisions(
    subtimestamp varchar(255) NOT NULL,
    username varchar(25) NOT NULL,
    prob_id int(11) NOT NULL,
    points int(3) NOT NULL,
    lang varchar(3) NOT NULL,
    PRIMARY KEY (subtimestamp),
);
```

Ο πίνακας tb_submisions δημιουργήθηκε για να αποθηκεύει τις υποβολές των λύσεων από τους χρήστες. Τα πεδία του πίνακα είναι ένας κωδικός (subtimestamp) που δημιουργείται αυτόματα με την υποβολή, το username, ο κωδικός του προβλήματος (prob_id), οι βαθμοί που έχει πάρει από τη συγκεκριμένη υποβολή (points) και το πεδίο lang που δηλώνει τη γλώσσα (Pascal, C/C++) που χρησιμοποιήθηκε για τη συγκεκριμένη υποβολή.

Όταν ένας χρήστης δημιουργήσει λογαριασμό (account) στο σύστημα τότε αυτόματα δημιουργείται και ένας φάκελος με το username του. Μέσα σε αυτόν το φάκελο θα αποθηκεύονται και οι υποβολές των ασκήσεων του. Για κάθε άσκηση δημιουργείται ένας φάκελος με το όνομα τον κωδικό της άσκησης. Μέσα σε αυτό τον φάκελο δημιουργούνται υποφάκελοι με όνομα το subtimestamp. Μέσα σε αυτούς τους υποφακέλους θα αποθηκεύονται οι υποβολές.

Όλα τα στοιχεία του πίνακα είναι απαραίτητα. Πρωτεύον κλειδί του πίνακα είναι το subtimestamp. Η μορφή που έχει το subtimestamp είναι YYYY-MM-DD-HH-MM-SS. Έχω θεωρήσει ότι δεν θα υπάρχουν υποβολές το ίδιο δευτερόλεπτο. Σε περίπτωση που παρατηρηθεί κάποιο πρόβλημα θα μπορούσα να προσθέσω σαν πρόθεμα στο subtimestamp και το username.

Πίνακας: tb_theory

```
Sql Statement
CREATE TABLE tb_theory(
    id int(11) NOT NULL,
    cat varchar(35) NOT NULL,
    filename varchar(255) NOT NULL,
    username varchar(20) NOT NULL,
    title varchar(20) NOT NULL,
    PRIMARY KEY (id),
);
```

Ο πίνακας αυτός δημιουργήθηκε για να αποθηκεύει τα στοιχεία που αφορούν τη θεωρεία που θα υποβληθεί από τους teachers/ administrators για κάθε κατηγορία προβλημάτων (π.χ. πίνακες, ταξινόμηση). Τα στοιχεία που αποθηκεύονται είναι η κατηγορία (cat), το όνομα του αρχείου (filename), ο χρήστης που ανέβασε το αρχείο (usename) και ο τίτλος (title) που θα εμφανίζεται στο σύστημα.

Όλα τα αρχεία της θεωρίας αποθηκεύονται στο φάκελο theory. Το όνομα του αρχείου αποτελείται από 2 προθέματα. Το πρώτο είναι το όνομα του χρήστη και το δεύτερο είναι το id του αρχείου. Με τον τρόπο αυτό επιτυγχάνεται η μοναδικότητα του ονόματος του αρχείου. Το όνομα αυτό αποθηκεύεται στο πεδίο filename.

Όλα τα πεδία του πίνακα είναι υποχρεωτικά. Το πεδίο id είναι το πρωτεύων κλειδί του πίνακα. Πρόκειται για έναν αύξων αριθμό.

Πίνακας: tb_problems_katigoria

```
Sql Statement
CREATE TABLE tb_problems_katigoria(
    katigoria varchar(35) NOT NULL,
    PRIMARY KEY (katigoria),
);
```

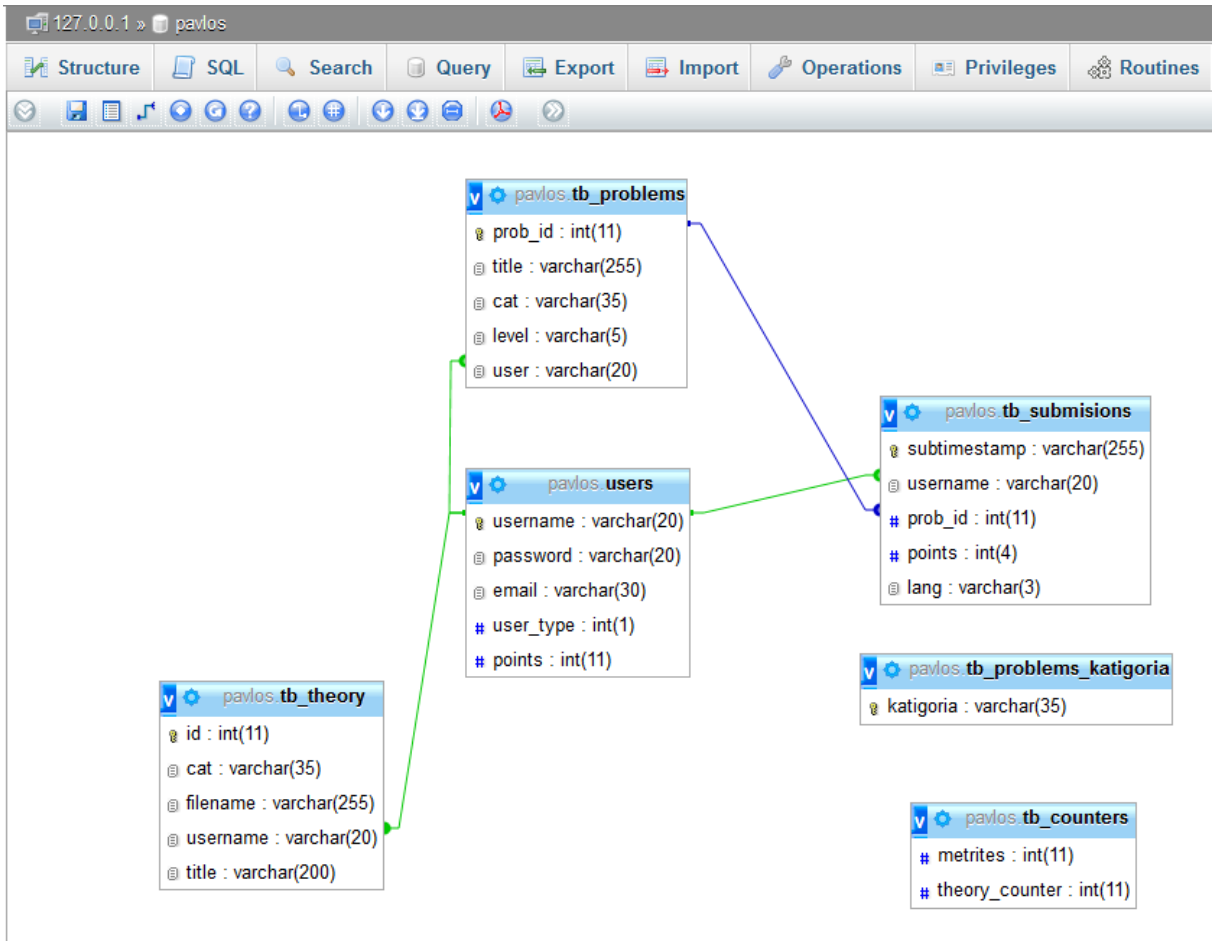
Ο πίνακας αυτός έχει αποθηκευμένες τις κατηγορίες των προβλημάτων. Θα τον χρησιμοποιώ για να «γεμίζω» τα μενού επιλογής.

Πίνακας: tb_counters

```
Sql Statement
CREATE TABLE tb_counters(
    metrites int(11) NOT NULL,
    theory_counters int(11) NOT NULL,
);
```

Μέσα σε αυτόν τον πίνακα θα αποθηκεύονται 2 μετρητές, που θα με βοηθήσουν στην υλοποίηση του συστήματος. Το πεδίο metrites αποθηκεύει το πλήθος των ασκήσεων που είναι αποθηκευμένες και το πεδίο theory_counters το πλήθος των αρχείων θεωρίας που έχουν υποβληθεί. Ο πίνακας δεν έχει πρωτεύον κλειδί.

Το σχήμα 16 παρουσιάζει τις σχέσεις μεταξύ των πινάκων.



Σχήμα 16: Σχέσεις μεταξύ των πινάκων

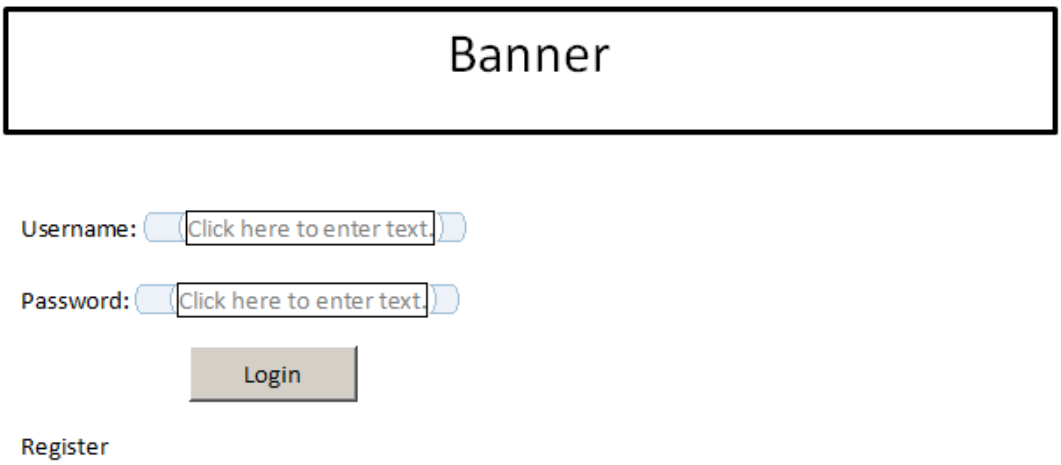
4.2 Σχεδιασμός Διαπαφών (Interface)

Ο σχεδιασμός του interface, δηλαδή των οθονών που θα βλέπει ο χρήστης είναι ένα πολύ σημαντικό στάδιο για την επιτυχία που θα έχει ένα σύστημα. Είναι επίσης πολύ σημαντικός παράγοντας δυσκολίας/ευκολίας για το στάδιο της υλοποίησης.

Ο στόχος είναι να σχεδιαστούν οι διαπαφές με τέτοιο τρόπο, ώστε το σύστημα να είναι εύχρηστο αλλά και να είναι εύκολες οι αλλαγές σε κώδικα. Θα προσπαθήσω να κρατήσω το γενικό πλαίσιο κοινό για όλους τους χρήστες και οι όποιες διαφοροποιήσεις επιβάλλονται να γίνονται με βάση αυτό το γενικό πλαίσιο. Επίσης θα χρησιμοποιήσω ένα αρχείο css (Cascading Style Sheet) για τα διάφορα στυλ (styles) που θα χρησιμοποιηθούν μέσα στον κώδικα HTML. Επίσης σκοπεύω να χρησιμοποιήσω την εντολή include για να περιλαμβάνω μέσα στις διάφορες σελίδες το κεντρικό μενού.

4.2.1 Διεπαφή Login

Η σελίδα του login θα είναι η πρώτη σελίδα την οποία θα βλέπει ο χρήστης. Πρέπει να είναι απλή και κατανοητή. Επίσης πρέπει να περιέχει σχετικό link για παραπομπή στη σελίδα της εγγραφής (register).



Σχήμα 17: Διεπαφή Login

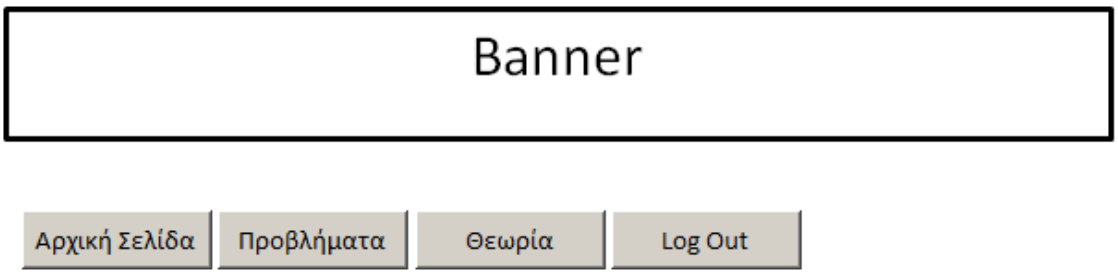
4.2.2 Διεπαφή Register

Σε περίπτωση που κάποιος δεν είναι εγγεγραμμένος στο σύστημα, θα μπορεί να το πράξει συμπληρώνοντας τα στοιχεία που φαίνονται στο πιο κάτω σχήμα. Οι οποιοιδήποτε περιορισμοί θα πρέπει να φαίνονται στην οθόνη.

Σχήμα 18: Διεπαφή Register

4.2.3 Μενού Μαθητή

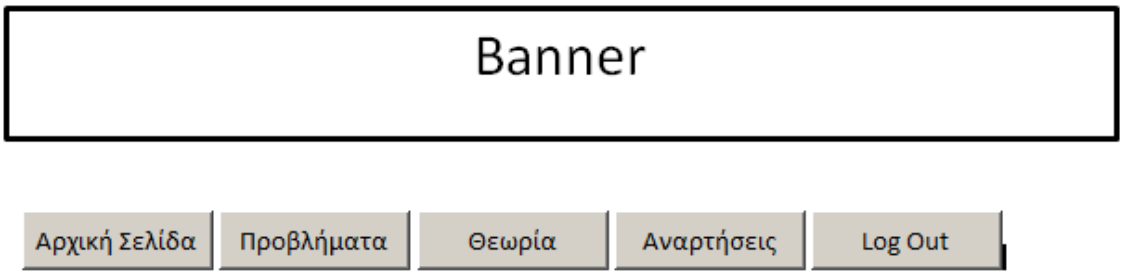
Το μενού του χρήστη μαθητή θα έχει τις λειτουργίες **Αρχική Σελίδα, Προβλήματα, Θεωρία, Logout.**



Σχήμα 19: Μενού χρήστη μαθητή

4.2.4 Μενού teacher

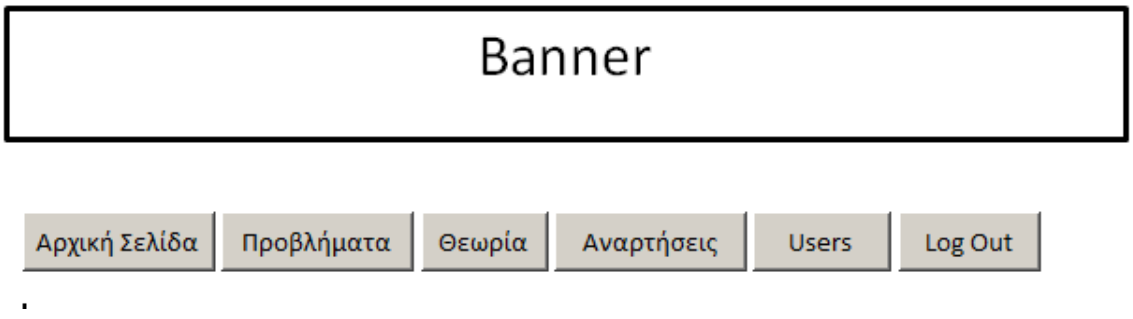
Το μενού του teacher θα έχει τις λειτουργίες **Αρχική Σελίδα, Προβλήματα, Θεωρία, Αναρτήσεις, Logout.**



Σχήμα 20: Μενού χρήστη teacher

4.2.5 Μενού administrator

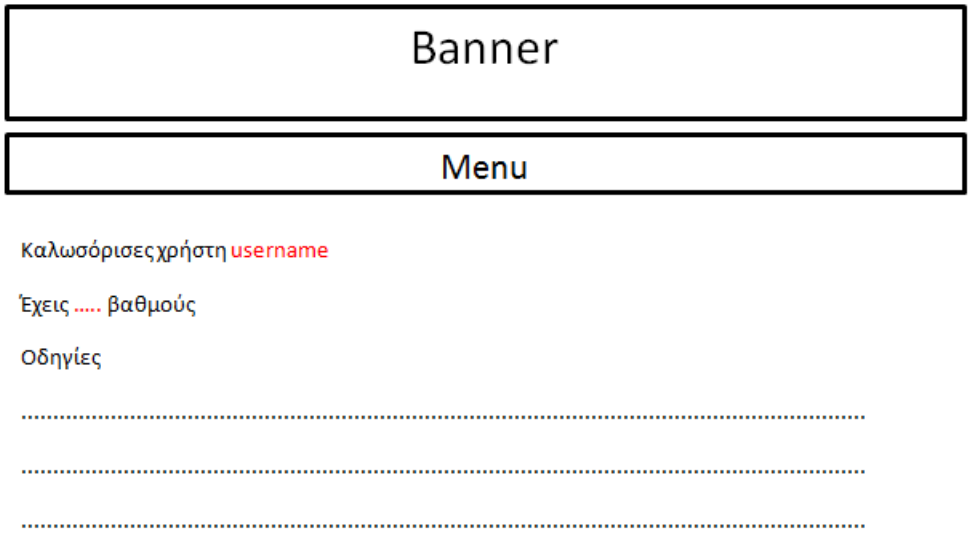
Το μενού του administrator θα έχει τις λειτουργίες **Αρχική Σελίδα, Προβλήματα, Θεωρία, Αναστήσεις, Users, Logout.**



Σχήμα 21: Μενού χρήστη administrtor

4.2.6 Αρχική Σελίδα

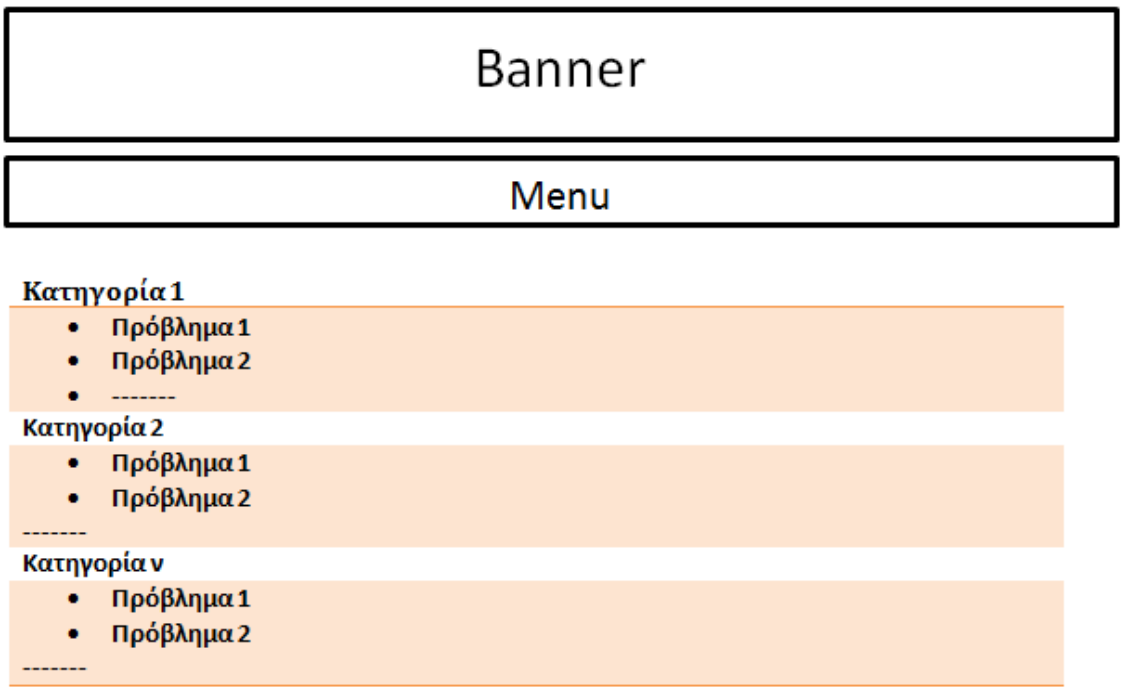
Η αρχική σελίδα θα παρουσιάζει το όνομα και τους βαθμούς έχει συγκεντρώσει μέχρι στιγμής ο χρήστης. Επίσης θα περιέχει ένα μικρό καλωσόρισμα και τους σκοπούς που επιτελεί το σύστημα.



Σχήμα 22: Οθόνη αρχικής σελίδας

4.2.7 Επιλογή Προβλήματος

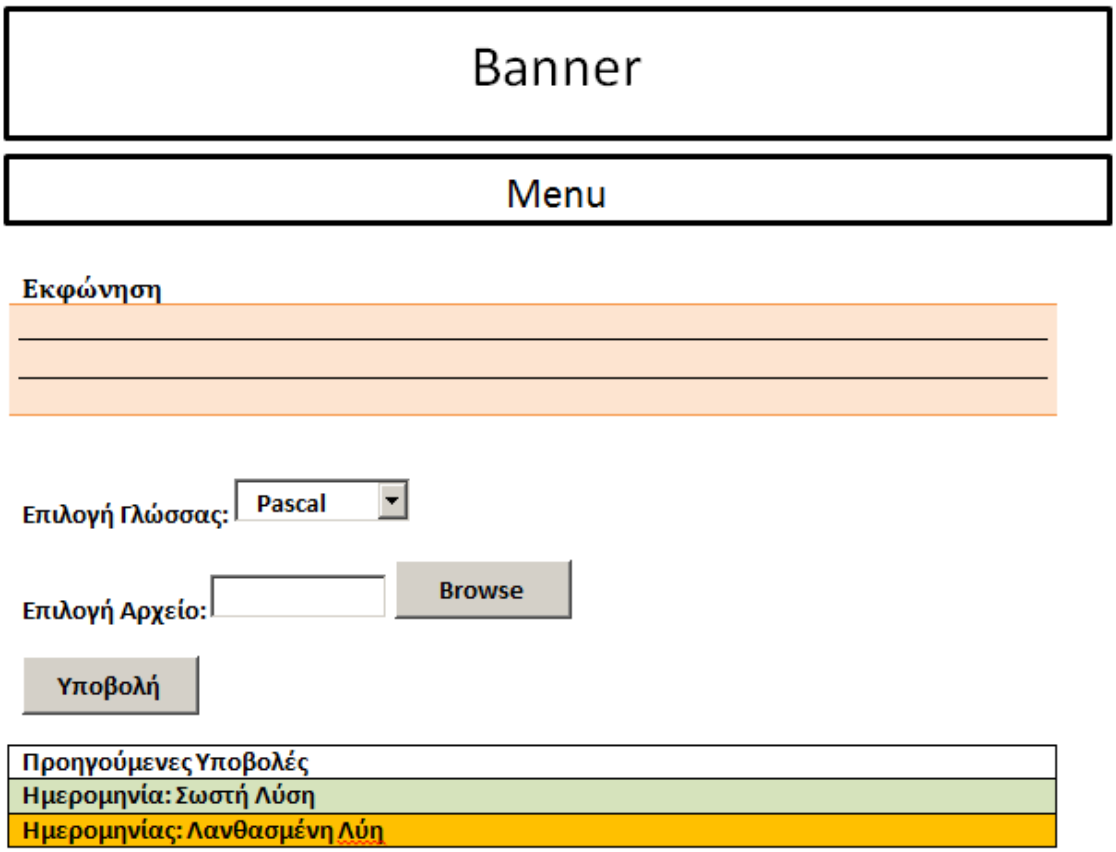
Σε αυτήν την οθόνη ο χρήστης θα μπορεί να επιλέγει το πρόβλημα που θέλει να λύσει. Τα προβλήματα θα είναι χωρισμένα σε κατηγορίες..



Σχήμα 22: Οθόνη επιλογής προβλήματος

4.2.8 Υποβολή Λύσης Προβλήματος

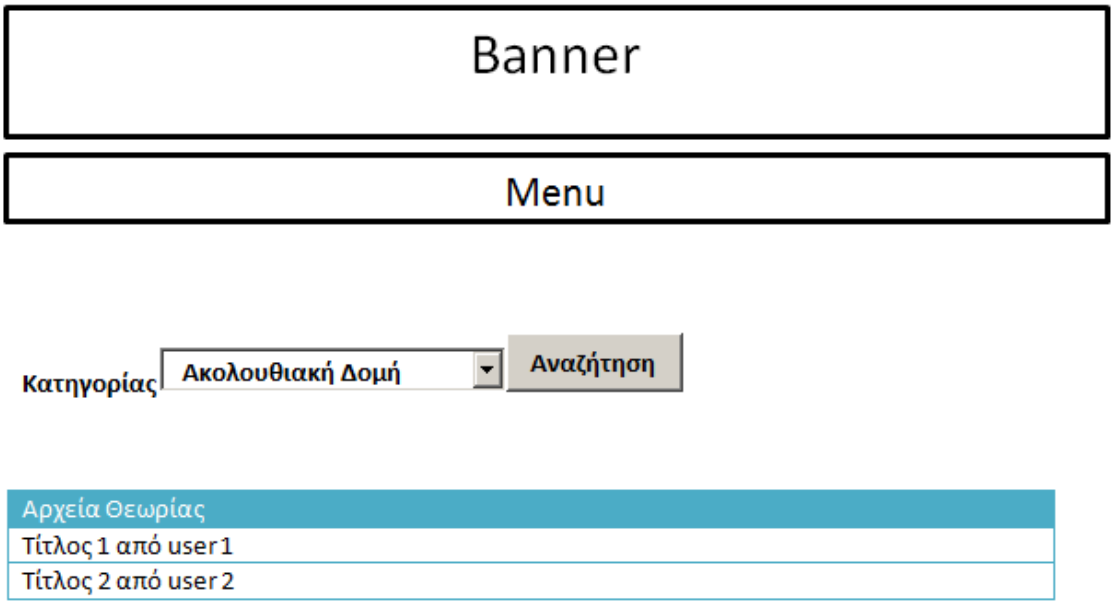
Αφού επιλέξει την άσκηση που θέλει να λύση ο χρήστης θα μεταβεί σε αυτή την οθόνη. Εδώ θα υπάρχει η εκφώνηση της άσκησης, η επιλογή της γλώσσας προγραμματισμού στην οποία θα δοθεί η λύση, ένα κουμπί για να γίνει upload η λύση καθώς και οι παλιές υποβολές του χρήστη σε αυτή την άσκηση.



Σχήμα 23: Οθόνη υποβολής λύσης προβλήματος

4.2.9 Θεωρία

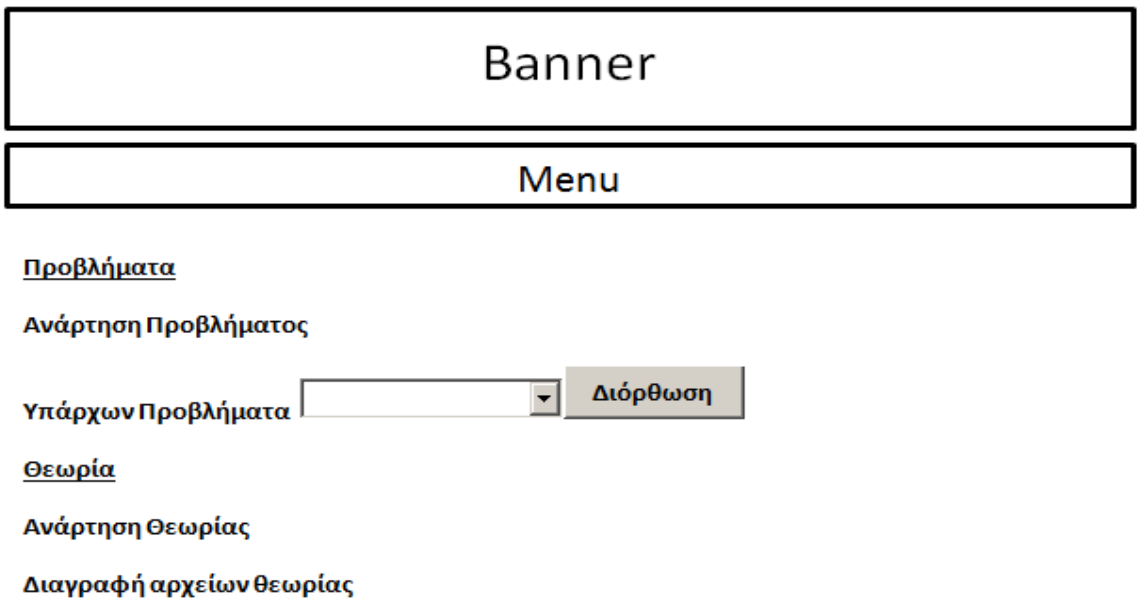
Όπως έχω ήδη αναφέρει οι χρήστες θα έχουν πρόσβαση σε θεωρία για κάθε κεφάλαιο. Η ανάρτηση των αρχείο της θεωρίας θα γίνεται από τους teachers και τους administrators. Στην αρχή ο χρήστης θα επιλέγει την κατηγορία και πατώντας το κουμπί αναζήτηση θα του παρουσιάζονται όλα τα αρχεία θεωρίας για την κατηγορία που επέλεξε. Για κάθε αρχείο θα εμφανίζεται ο τίτλος του και ο χρήστης που το ανάρτησε.



Σχήμα 24: Οθόνη αρχείων θεωρίας

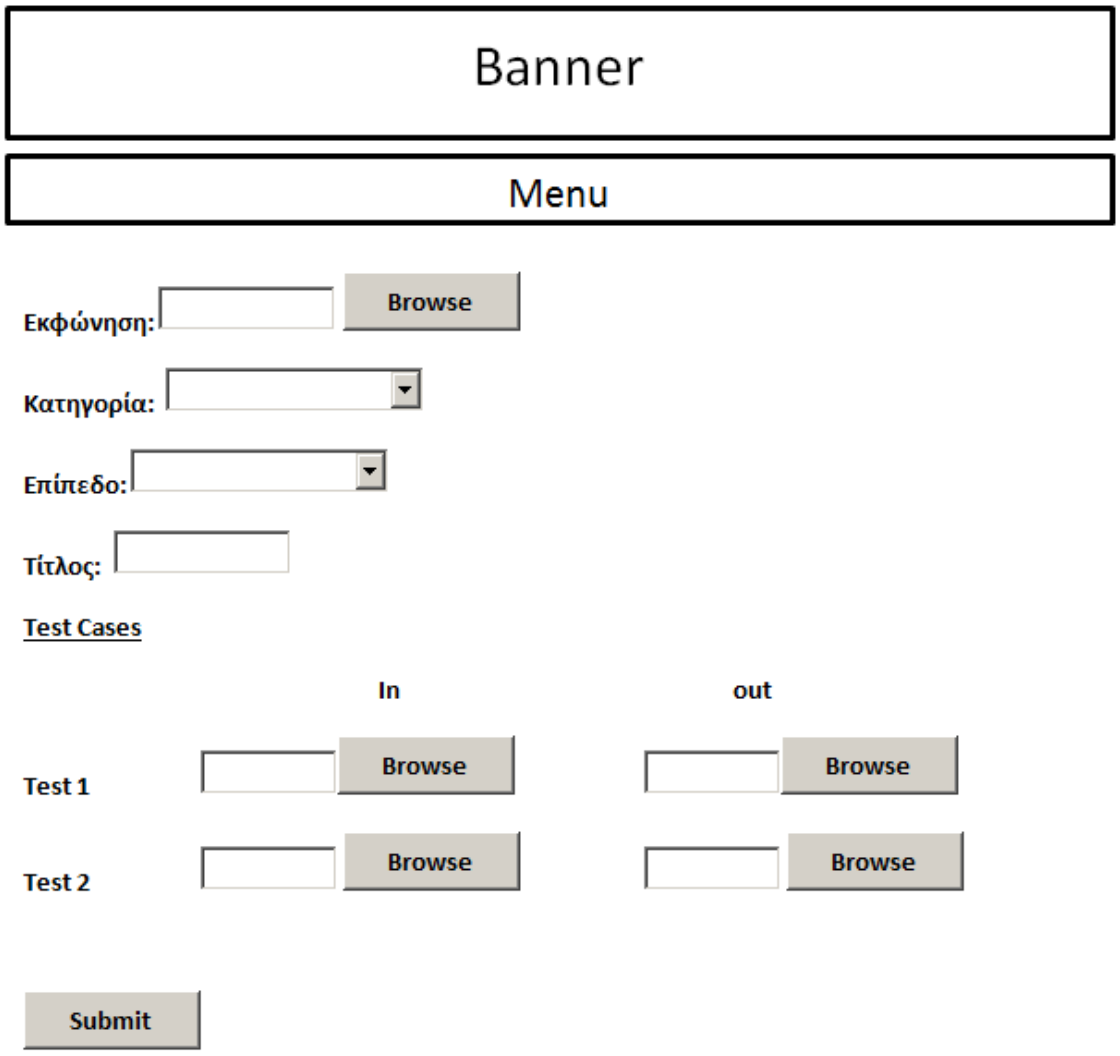
4.2.10 Αναρτήσεις

Το μενού «Αναρτήσεις» εμφανίζεται μόνο για τους χρήστες Teachers/ Administrators. Μέσα από αυτή τη σελίδα οι χρήστες θα μπορούν να αναρτήσουν προβλήματα/ αρχεία θεωρίας, να διορθώσουν παλιές αναρτήσεις προβλημάτων και να διαγράψουν αρχεία θεωρίας. Οι Teachers θα έχουν πρόσβαση μόνο στα δικά τους αρχεία ενώ οι administrators σε όλα τα αρχεία.



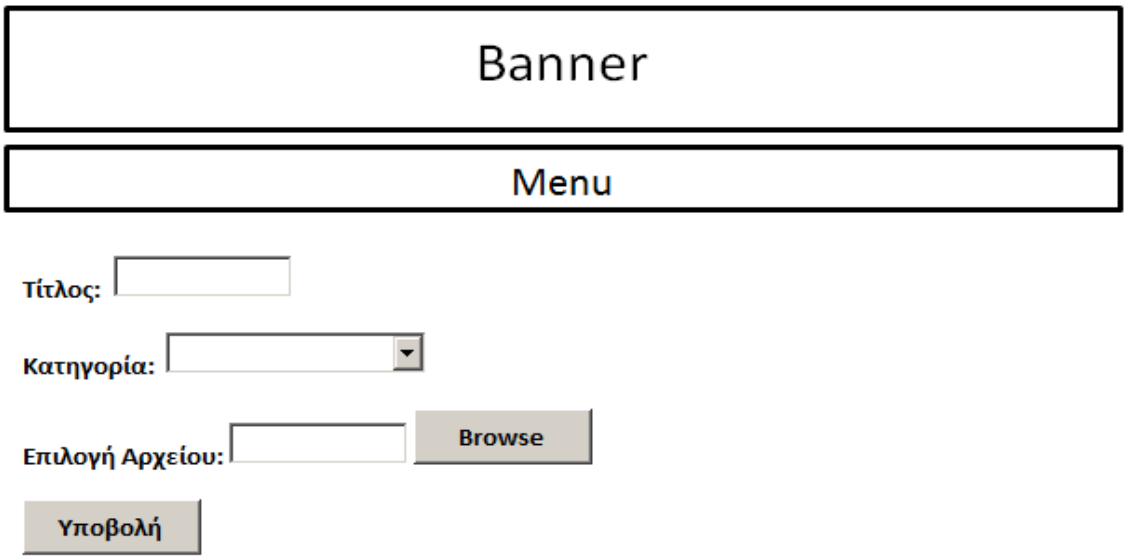
Σχήμα 25: Οθόνη αναρτήσεων

Πατώντας το σύνδεσμο «Ανάρτηση Προβλήματος» ο χρήστης θα μεταβαίνει στη σχετική σελίδα. Εκεί θα πρέπει να καταχωρεί τον τίτλο, να επιλέγει την κατηγορία και το επίπεδο και να ανεβάζει την εκφώνηση σε μορφή txt. Επίσης θα πρέπει να ανεβάζει τα test cases. Για κάθε test case θα πρέπει να ανεβάζει την είσοδο και την αναμενόμενη έξοδο.



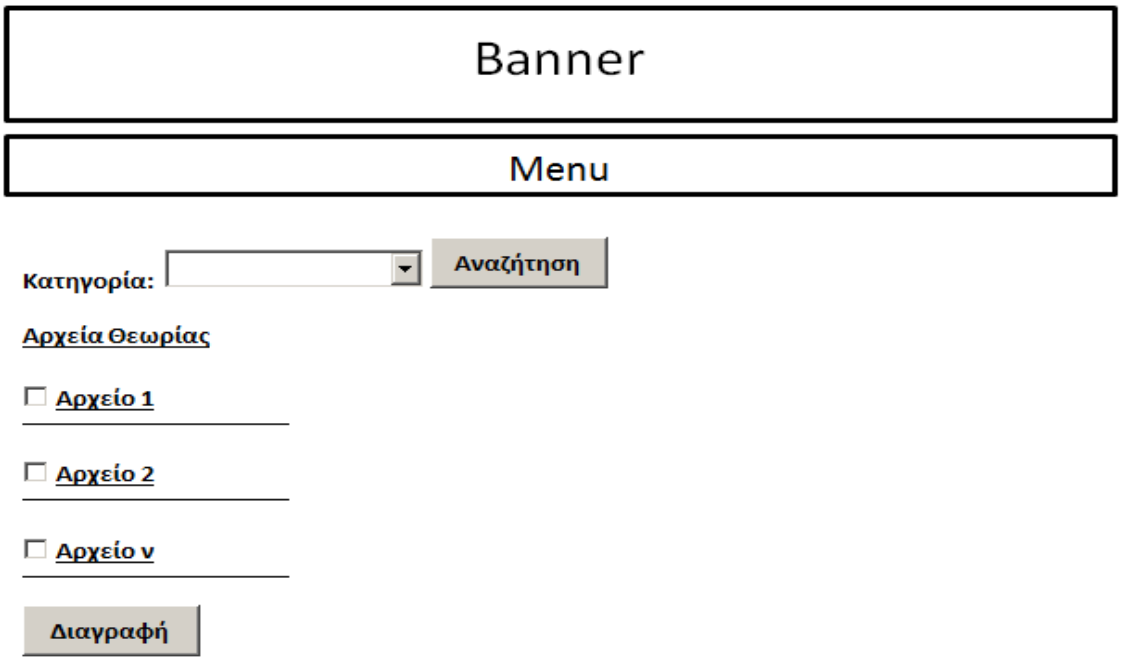
Σχήμα 26: Οθόνη ανάρτησης προβλήματος

Όσο αφορά την ανάρτηση αρχείου θεωρίας, χρήστης θα γράφει τον τίτλο, θα επιλέγει την κατηγορία και θα ανεβάζει το αρχείο.



Σχήμα 27: Οθόνη ανάρτησης θεωρίας

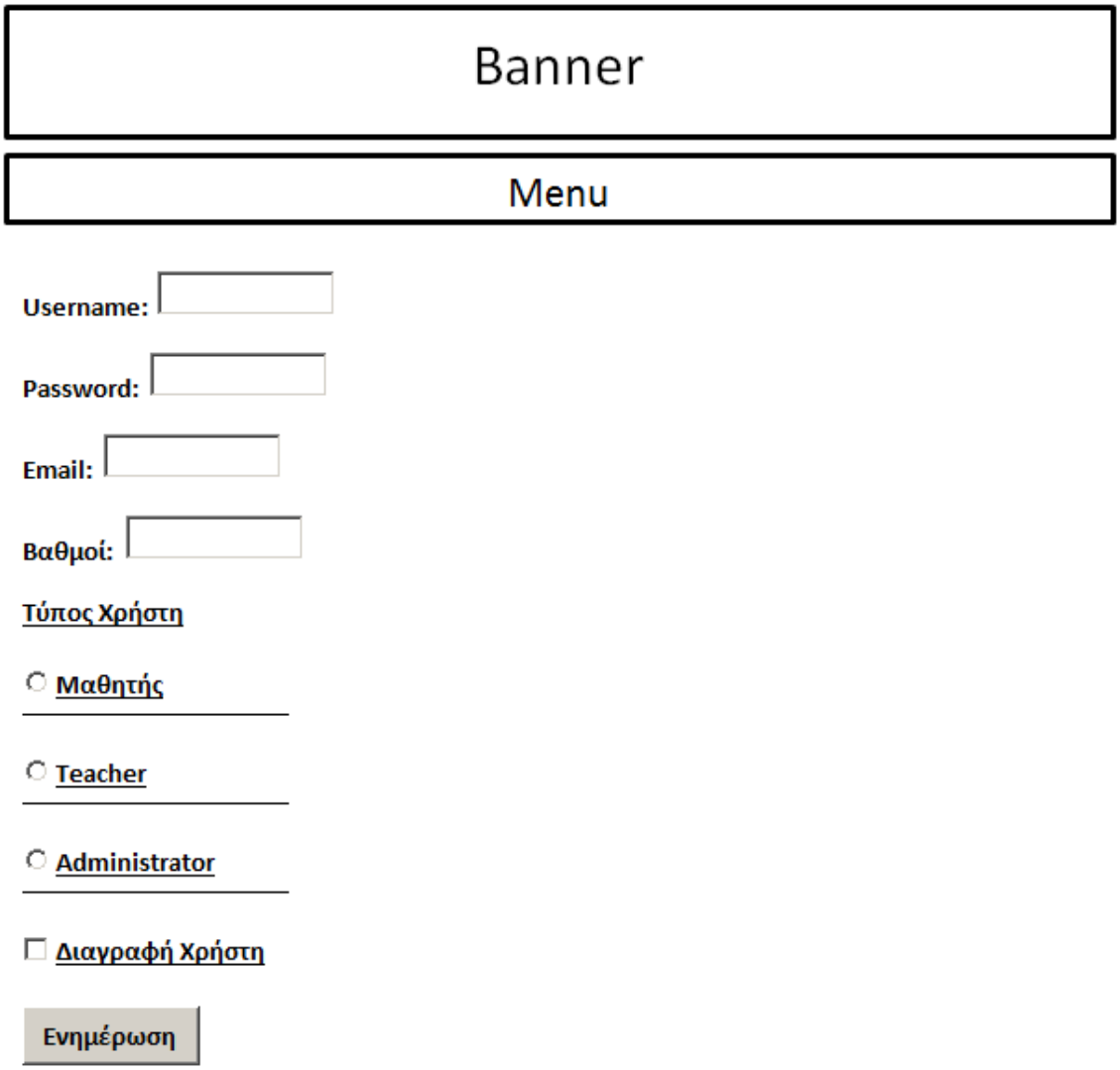
Τέλος όσο αφορά τη διαγραφή των αρχείων θεωρίας, ο χρήστης θα επιλέγει την κατηγορία και θα εμφανίζονται όλα τα αρχεία στη συγκεκριμένη κατηγορία. Δίπλα από κάθε αρχείο θα εμφανίζεται ένα check box. Θα σημειώνουμε τα αρχεία που θέλουμε να διαγραφούν και θα πατάμε το αντίστοιχο κουμπί. Να σημειώσω και πάλι ότι οι teachers έχουν μόνο πρόσβαση στα δικά τους αρχεία ενώ οι administrators σε όλα.



Σχήμα 28: Οθόνη διαγραφής αρχείων θεωρίας

4.2.11 Users

Σε αυτή την οθόνη έχει πρόσβαση μόνο ο χρήστης του τύπου administrator. Μέσα από αυτήν την οθόνη ο χρήστης θα μπορεί να διορθώσει ή και να διαγράψει κάποιον από τους χρήστες. Όσο αφορά τη διόρθωση θα μπορεί να γίνεται για τα στοιχεία username, password, email, βαθμοί και τύπος χρήστη.



Σχήμα 29: Οθόνη αλλαγής ρυθμίσεων χρηστών

4.3 Αρχεία στον server

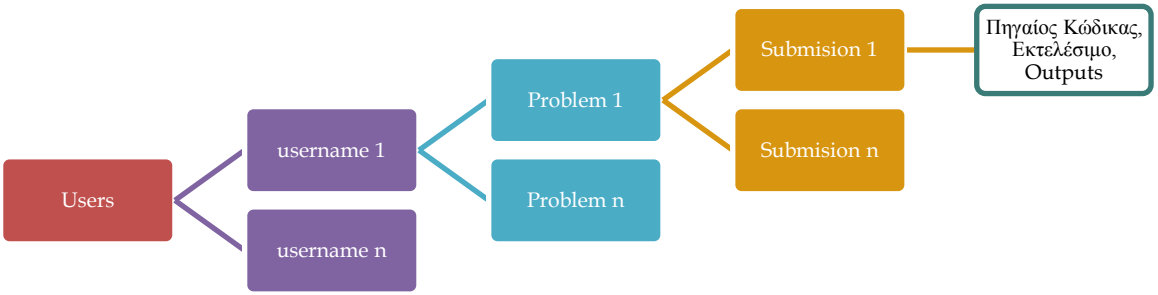
Όπως αναφέρθηκε και προηγουμένως, ορισμένα από τα αρχεία των χρηστών αποθηκεύονται στον server. Θα μπορούσαμε να χωρίσουμε τα αρχεία σε 3 κατηγορίες. Στα αρχεία που αφορούν τους χρήστες, αρχεία που αφορούν τις ασκήσεις και αρχεία που αφορούν τη θεωρία.

4.3.1 Αρχεία χρηστών

Με την εγγραφή του χρήστη δημιουργείται αυτόματα μέσα στο φάκελο users, ένας φάκελος με το username του χρήστη. Μέσα σε αυτόν το φάκελο δημιουργούνται υποφακέλοι για κάθε άσκηση που δοκιμάζει ο χρήστης. Μέσα σ το φάκελο της άσκησης δημιουργείται ένας φάκελος για κάθε υποβολή. Στο φάκελο της υποβολής θα υπάρχει:

- Ο πηγαίος κώδικας που ανέβασε ο χρήστης
- Το εκτελέσιμο που δημιουργείται από το σύστημα με βάση τον πηγαίο κώδικα που έχει υποβάλει ο χρήστης.
- Τα αχρεία εξόδου που δημιουργούνται με βάση το εκτελέσιμο και τα αρχεία εισόδου της άσκησης.

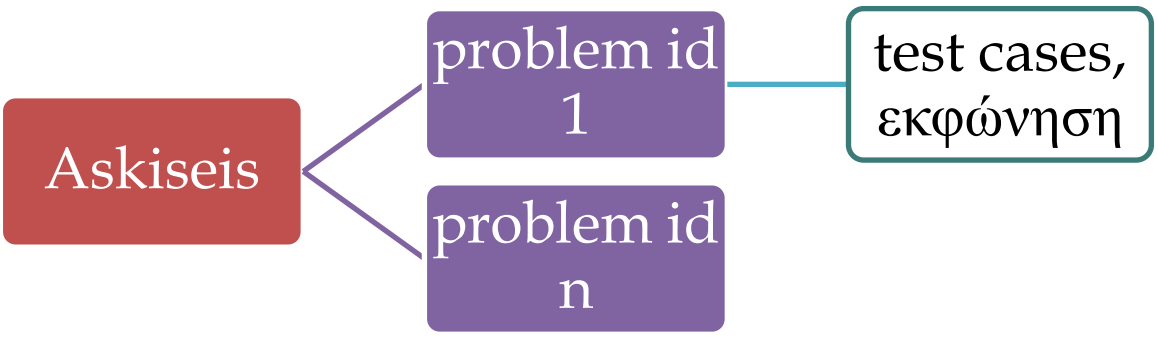
Στο πιο κάτω σχήμα παρουσιάζεται η δομή των φακέλων όπως εξηγήθηκε πιο πάνω.



Σχήμα 30: Διάγραμμα φακέλων χρήστη

4.3.2 Αρχεία Ασκήσεων

Στον αρχικό φάκελο του συστήματος υπάρχει ένας φάκελος με όνομα **askiseis.** Κάθε φορά που κάποιος ανεβάζει μια νέα άσκηση, δημιουργείται μέσα σε αυτόν το φάκελο, ένας υποφάκελος με όνομα το id της άσκησης. Σε αυτόν υπάρχουν αποθηκευμένα τα test cases (inputs-outputs) και η εκφώνηση της άσκησης. Στο παρακάτω σχήμα παρουσιάζεται η δομή των φακέλων.



Σχήμα 31: Διάγραμμα φακέλων ασκήσεων

4.3.3 Αρχεία Θεωρίας

Όλα τα αρχεία θεωρίας αποθηκεύονται μέσα στο φάκελο **theory**. Το κάθε αρχείο ονομάζεται ως εξής: το πρώτο πρόθεμα του ονόματος του αρχείου είναι το username του χρήστη που το ανέβασε και το δεύτερο πρόθεμα είναι το id του αρχείου (ένας αύξων αριθμός) που αποθηκεύεται στη βάση δεδομένων. Για παράδειγμα αν ο χρήστης **user** ανεβάσει ένα αρχείο θεωρίας το οποίο θα έχει σαν id τον αριθμό 6 τότε το όνομα του αρχείου θα είναι **user6**.

Κεφάλαιο 5
Υλοποίηση

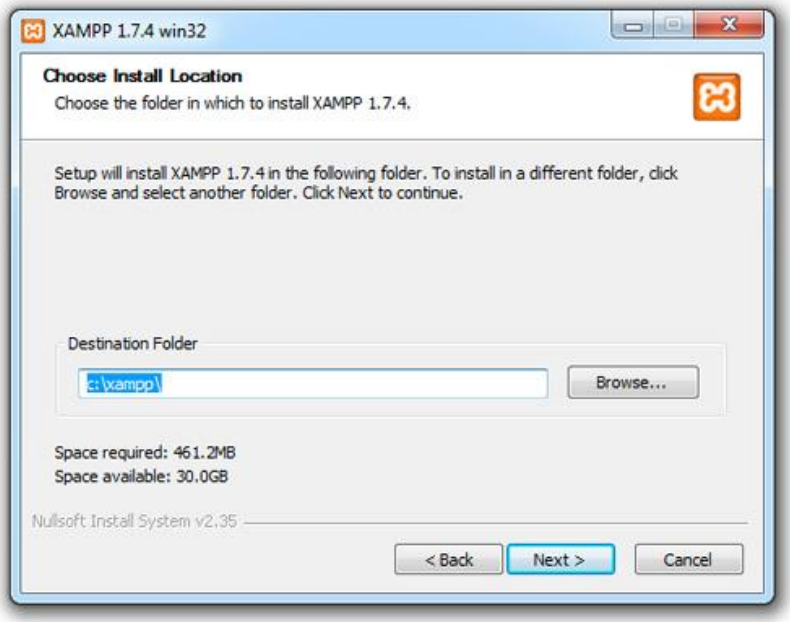
Στο κεφάλαιο αυτό θα παρουσιάσω τον τρόπο με το οποίο υλοποιήθηκε το σύστημα. Θα δείξω την εγκατάσταση και τη χρήση του XAMPP, τον τρόπο με τον οποίο έγινε η σύνδεση του συστήματος με τη βάση δεδομένων, την εκτέλεση διαφόρων λειτουργιών του συστήματος και το μηχανισμό με τον οποίο γίνεται αξιολόγηση της άσκησης. Θα παραθέσω μόνο κάποια κομμάτια του κώδικα τα οποία θεωρώ σημαντικά. Όλα τα αρχεία είναι διαθέσιμα για οποιαδήποτε χρήση από το Ανοικτό Πανεπιστήμιο Κύπρου.

Τέλος θα παρουσιάσω τις ασκήσεις και τη θεωρία που έχω βάλει στο σύστημα στις κατηγορίες ακολουθιακή δομή, δομή διακλάδωσης, επαναληπτική δομή, πίνακες, στοίβες και γράφοι. Σε κάθε άσκηση θα παρουσιάζονται τα test cases ο λόγος επιλογής τους.

5.1 Εγκατάσταση του XAMPP

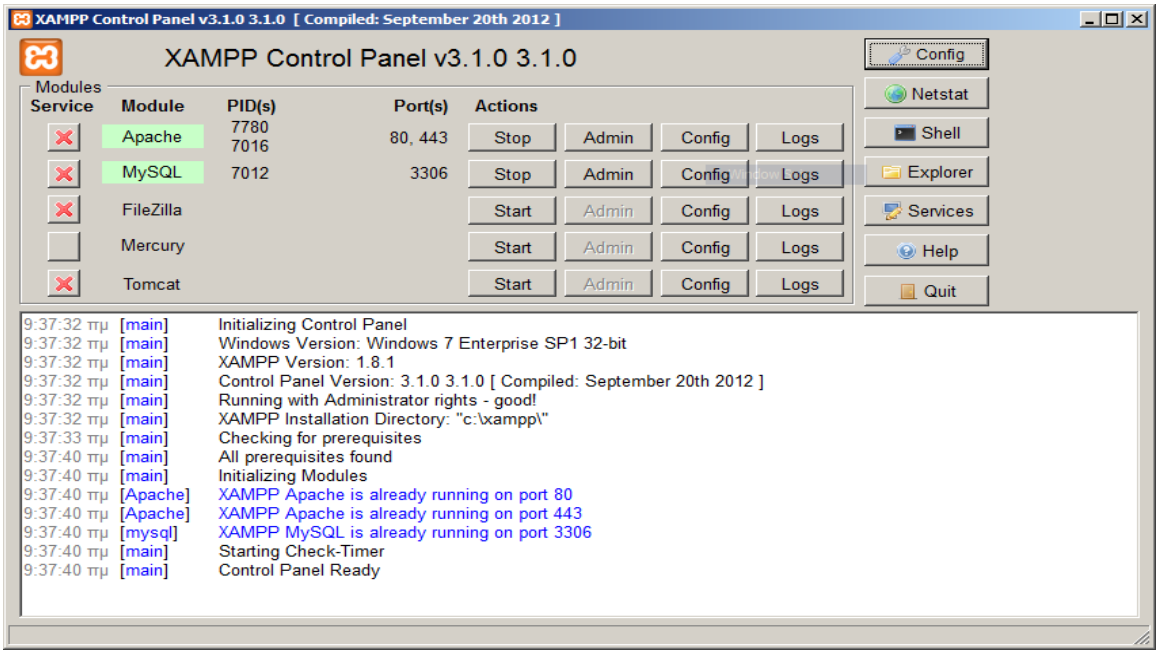
Το XAMMP είναι ένας εύκολος και δωρεάν τρόπος να εγκαταστήσει κάποιος τον Apache μαζί με τη MySQL και το PHP. Υπάρχουν εκδόσεις του XAMPP για Windows (αυτή που χρησιμοποίησα), για Linux, Mac OS και Solaris.

Ο πιο εύκολος τρόπος να εγκαταστήσει κάποιος το XAMPP είναι να χρησιμοποιήσει το σχετικό installer.



Σχήμα 31: Installer XAMPP

Στη συνέχεια το μόνο που έχει να κάνει είναι ξεκινήσει τα processes του Apache και της MySQL.



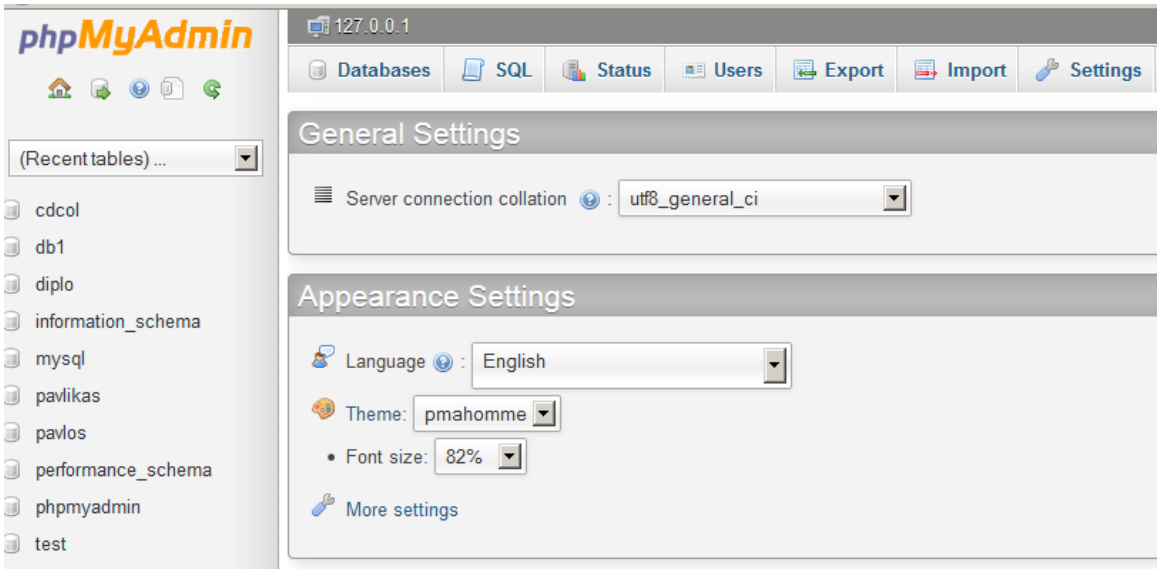
Σχήμα 32: Αρχική οθόνη XAMPP

Η έκδοση του XAMPP που έχω χρησιμοποιήσει είναι η 1.8.1 και σε αυτή περιλαμβάνονται:

- Apache HTTP Server Version 2.4
- MySQL 5.6
- PHP 5.4

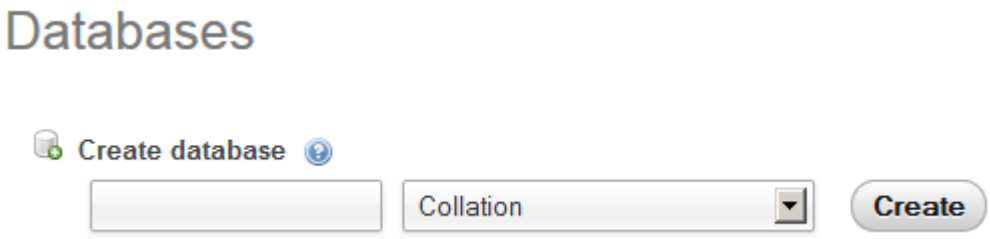
5.2 Σύνδεση με τη βάση δεδομένων

Για τη διαχείριση της βάσης δεδομένων χρησιμοποίησα το εργαλείο phpMyAdmin που υπήρχε εγκατεστημένο μέσα στο XAMPP. Ένας εναλλακτικός τρόπος είναι μέσα από το πρόγραμμα MySQL Workbench

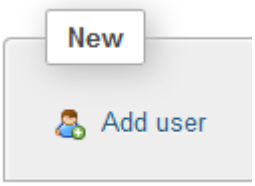


Σχήμα 33: Αρχική οθόνη MySQL

Δημιούργησα μια βάση δεδομένων με το όνομα pavlos και έναν νέο user με όνομα puser.



Σχήμα 34: Δημιουργία βάσης δεδομένων



Σχήμα 35: Δημιουργία νέου χρήστη

Το επόμενο βήμα ήταν να δώσω πλήρη δικαιώματα (ALL PRIVILEGES) στον puser, ώστε να μπορεί να εκτελεί όλες τις λειτουργίες πάνω στη βάση δεδομένων. Η δημιουργία των πινάκων έγινε με τη χρήση των SQL Statements που παρουσιάστηκαν στο κεφάλαιο της σχεδίασης. Τέλος σχεδιάστηκαν οι σχέσεις μεταξύ των πινάκων.

Η σύνδεση της βάσης δεδομένων με το σύστημα έγινε την παρακάτω δήλωση στη php [03].

`$myDB = mysqli_connect("localhost","puser","puser","pavlos");` όπου:

- Localhost είναι το όνομα του host. Θα μπορούσε να είναι και μια διεύθυνση IP
- Puser το όνομα του user
- Puser το password του user
- Pavlos το όνομα της βάσης δεδομένων.

Η συνάρτηση mysqli_connect επιστρέφει στη μεταβλητή $myDB ένα αντικείμενο που αναπαριστά τη σύνδεση με τη MySQL.

5.3 Ερωτήματα

Στο σύστημα υπάρχουν τεσσάρων(4) ειδών ερωτήματα. Υπάρχουν ερωτήματα επιλογής (Select), εισαγωγής (Insert), διαγραφής (Delete) και ενημέρωσης (Update). Για κάθε ένα από τα ερωτήματα δημιουργείται ένα SQL statement το οποίο συνήθως περιέχει και μεταβλητές του συστήματος και στη συνέχεια αυτό εκτελείται με την κατάλληλη εντολή. [06]

5.3.1 Ερωτήματα Επιλογής

Τα ερωτήματα επιλογής χρησιμοποιούνται για να επιλέξουν συγκεκριμένα δεδομένα από τη βάση. Για το λόγο αυτό χρησιμοποιείται η εντολή **Select.** Η γενική σύνταξη της εντολής είναι:

```
SELECT column_name,column_name FROM table_name;
```

Παρακάτω παρουσιάζω μερικές περιπτώσεις που έχω χρησιμοποιήσει ερωτήματα επιλογής.

Σύνδεση χρήστη (Login)

Όταν ο χρήστης θέλει να συνδεθεί στο σύστημα θα εισάγει το username και το password του. Το σύστημα ψάχνει στον πίνακα users αν υπάρχει χρήστης με τα στοιχεία που έδωσε ο χρήστης. Για το λόγο αυτό δημιουργείται το ερώτημα [03][05]:

```
$query = "select * from users where username ='".$username."' and password = '".$pass."'";

$result = mysqli_query($myDB, $query);
```

Αν το αποτέλεσμα του ερωτήματος είναι το κενό σύνολο τότε σημαίνει ότι δε βρέθηκε χρήστης με αποτέλεσμα η σύνδεση να μην είναι δυνατή.

```
if (!($row = mysqli_fetch_array($result))){

        session_destroy();

        include 'login.php';

        echo "<p>Λανθασμένο login. Ξαναδοκιμάστε</p>";

        exit;     }
```

Παρουσίαση Προβλημάτων

Η σελίδα παρουσίασης των προβλημάτων δημιουργείται δυναμικά με τη χρήση ερωτημάτων επιλογής. Στην αρχή δημιουργείται ένα σύνολο με όλες τις κατηγορίες των προβλημάτων. Στη

συνέχεια επιλέγονται και παρουσιάζονται σε έναν πίνακα τα προβλήματα για κάθε μια από τις κατηγορίες. Το παρακάτω τμήμα του κώδικα παρουσιάζει τη σχετική λειτουργία [03][05].

```
$sql="Select * from tb_problems_katigoria";

$result1 = mysqli_query($myDB, $sql);

while($row1 = mysqli_fetch_array($result1)){

        echo "<tr><td bgcolor='#33CCFF'>".$row1['katigoria']."</td></tr>";

        $query = "select * from tb_problems where cat='".$row1['katigoria']."'";

        $result = mysqli_query($myDB, $query);

        while($row = mysqli_fetch_array($result)){

                $id=$row['prob_id'];

echo "<tr><td><a href='problems1.php?id=".$id."'>"; ?> <?php echo $row['title']; ?></td></tr><?php

        }//while provlimawn

}//while katigoriwn
```

5.3.2 Ερωτήματα Εισαγωγής

Τα ερωτήματα εισαγωγής χρησιμοποιούνται για να εισάγουν δεδομένα στους πίνακες της βάσης. Για το λόγο αυτό χρησιμοποιείται η εντολή **Insert.** Η γενική σύνταξη της εντολής είναι:

```
INSERT INTO table_name (column1,column2,column3,...)
VALUES (value1,value2,value3,...);
```

Παρακάτω παρουσιάζω μερικές περιπτώσεις που έχω χρησιμοποιήσει ερωτήματα εισαγωγής.

Εισαγωγή νέου χρήστη

Όταν ο ένας νέος χρήστης κάνει εγγραφή στο σύστημα (register) τα στοιχεία του ελέγχονται για την ορθότητα τους και στη συνέχεια καταχωρούνται στον πίνακα users. Αξίζει ότι το password του κάθε χρήστη κωδικοποιείται με τον αλγόριθμο **sha1.**

```
$query = "INSERT INTO users (username,password,email,user_type) VALUES
('".$username."','".sha1($pass1)."','".$email."',1);";

$result = mysqli_query($myDB, $query);
```

Ανάρτηση Προβλήματος

Η ανάρτηση των προβλημάτων χωρίζεται ουσιαστικά σε δύο μέρη. Το πρώτο αφορά τη φύλαξη της εκφώνησης και των test cases στο φάκελο του προβλήματος και το δεύτερο τη φύλαξη των πληροφοριών (id, τίτλος, κατηγορία, επίπεδο, χρήστης που του αναρτά) του προβλήματος στη βάση δεδομένων. Η φύλαξη των πληροφοριών στον πίνακα **tb_problems** γίνεται με τις παρακάτω εντολές.

```
$sql="INSERT INTO tb_problems(prob_id,title,cat,level,user) VALUES
(".$id.",'".$title."','".$katigoria."','".$level."','".$u."')";

mysqli_query($myDB,$sql);
```

5.3.3 Ερωτήματα Διαγραφής

Τα ερωτήματα εισαγωγής χρησιμοποιούνται για να διαγράψουν δεδομένα από τους πίνακες της βάσης. Για το λόγο αυτό χρησιμοποιείται η εντολή **Delete.** Η γενική σύνταξη της εντολής είναι:

```
DELETE FROM table_name WHERE some_column=some_value;
```

Παρακάτω παρουσιάζω μερικές περιπτώσεις που έχω χρησιμοποιήσει ερωτήματα διαγραφής.

Διαγραφή Χρήστη

Η λειτουργία αυτή είναι επιτρεπτή στον administrator και γίνεται μέσα από το μενού **users**.

```
$query="Delete from users where username='".$username."'";
```

```
$result = mysqli_query($myDB, $query);
```

Διαγραφή αρχείων θεωρίας

Δυνατότητα διαγραφής αρχείων θεωρίας έχουν οι χρήστες teacher και administrator. Αυτό επιτυγχάνεται με τις πιο κάτω εντολές.

```
$sql="Delete from tb_theory where id=".$thid;

$result = mysqli_query($myDB, $sql);
```

5.3.4 Ερωτήματα Ενημέρωσης

Τα ερωτήματα εισαγωγής χρησιμοποιούνται για να ενημερώσουν δεδομένα που βρίσκονται στους πίνακες της βάσης. Για το λόγο αυτό χρησιμοποιείται η εντολή **Update.** Η γενική σύνταξη της εντολής είναι:

```
UPDATE table_name
SET column1=value1,column2=value2,...
WHERE some_column=some_value;
```

Παρακάτω παρουσιάζω μερικές περιπτώσεις που έχω χρησιμοποιήσει ερωτήματα ενημέρωσης.

Αλλαγή στοιχείων χρήστη

Η λειτουργία αυτή είναι επιτρεπτή μόνο στον administrator και του επιτρέπει να αλλάξει κάποιες από τις πληροφορίες του χρήστη (π.χ. τύπος χρήστη)

```
$query="UPDATE users SET username='".$username."', password='".sha1($pass)."',
email='".$email."', user_type=".$user_type.", points=".$points." WHERE
username='".$username."'";

$result = mysqli_query($myDB, $query);
```

Διόρθωση προβλήματος

Μέσα από τη λειτουργία διόρθωσης προβλήματος, οι χρήστες teacher και administrator μπορούν να διορθώσουν τις πληροφορίες ενός προβλήματος (π.χ. τίτλος). Για να γίνει αυτό έχω γράψει τις παρακάτω εντολές.

```
$sql="Update tb_problems set title='".$title."', cat='".$c."', level='".$lev."' where prob_id=".$id;

mysqli_query($myDB,$sql);
```

5.4 Αυτόματη αξιολόγηση της άσκησης

Το κομμάτι της αυτόματης αξιολόγησης της άσκησης ήταν ίσως το πιο δύσκολο να υλοποιηθεί. Μετά από έρευνα βρήκα ότι υπήρχαν διάφορες στρατηγικές αντιμετώπισης του προβλήματος. Η δημοφιλέστερη ήταν η εξής: Ο χρήστης ανεβάζει τα αρχεία του στο σύστημα και αυτό τα τοποθετεί σε μια ουρά (queue). Πάνω στον server υπάρχει μια διαδικασία (watcher process) που τρέχει συνεχώς και ψάχνει για αρχεία στην ουρά. Αν υπάρχουν αρχεία στην ουρά τότε η διαδικασία εκτελεί τον κώδικα και παράγει τα αποτελέσματα με βάση τα test cases της άσκησης. Η διαδικασία μπορεί να είναι ένα chron job ή ένα daemon.

Αποφάσισα να χρησιμοποιήσω μια παραλλαγή του πιο πάνω τρόπου επίλυσης του προβλήματος. Αντί να τοποθετώ τα αρχεία σε μια ουρά και να περιμένω το watcher process να αναλάβει τη βαθμολόγηση της άσκησης, δημιουργώ για κάθε υποβολή μια νέα διαδικασία η οποία είναι υπεύθυνη για την αξιολόγηση της άσκησης και έχει διάρκεια ζωής μόνο όσο εκτελείται η άσκηση. Το όνομα της διαδικασίας είναι το username του χρήστη. Ο τρόπος αυτός έχει προφανή πλεονεκτήματα και μειονεκτήματα.

Τα πλεονεκτήματα του είναι τα εξής:

- Το σύστημα τη περισσότερη ώρα είναι αδρανές, οπότε η συνεχής λειτουργία του watcher process θα κατανάλωνε πόρους χωρίς λόγους. Με τον τρόπο που χρησιμοποίησα το σύστημα καταναλώνει πόρους όποτε τους χρειάζεται.

- Τα αρχεία του χρήστη δεν χρειάζεται να περιμένουν στην ουρά, αλλά εκτελούνται αμέσως.

Τα μειονεκτήματα του είναι τα εξής:

- Αν καταφτάσουν στο σύστημα πολλές υποβολές τότε θα έχουμε υπερφόρτωση του συστήματος.

- Είναι πιο δύσκολη η υλοποίηση του.

Θεωρώ ότι για το μέγεθος της Κύπρου και με τη χρήση ενός καλού συστήματος, τα πλεονεκτήματα υπερτερούν των μειονεκτημάτων.

Όπως είπα και πριν η δημιουργία της διαδικασίας ήταν το πιο δύσκολο κομμάτι που έπρεπε να υλοποιήσω. Ήταν κάτι για το οποίο δεν βρίσκεις πολλές σελίδες να διαβάσεις μιας και δεν είναι κάτι που θέλει να δημιουργήσει κάποιος πολύ συχνά. Η λύση στο πρόβλημα μου ήρθε μέσα από τον πειραματισμό μου με τα batch files. Το batch file είναι βασικά ένα αρχείο που περιέχει εντολές που εκτελούνται από τον command interpreter. Δηλαδή πρόκειται για command line εντολές. Ο βασικός λόγος που κατάφυγα στα batch files είναι ότι τόσο οι gcc/g++ (compiler για C/C++) όσο και ο fpc (compiler για τη free Pascal) δέχονται command line εντολές.

Τα βήματα αξιολόγησης της άσκησης είναι τα ακόλουθα:

1. Ο πηγαίος κώδικας γίνεται compile και εκτελείται το εκτελέσιμο, ανάλογα με τη γλώσσα που έχει επιλέξει ο χρήστης. Το κομμάτι του κώδικα υπεύθυνο για αυτό είναι το ακόλουθο.

```
$file = 'test.bat';

if($ext=="cpp"){

 move_uploaded_file($_FILES["file"]["tmp_name"],$st1."/".$pid.".cpp");

$current = "C:\Dev-Cpp\bin\g++ users\\". $u."\\". $pid."\\".$tstamp."\\". $pid.".cpp -o users\\".
$u."\\". $pid."\\".$tstamp."\\". $u .".exe 2> users\\". $u."\\". $pid."\\".$tstamp."\\output.txt \n";}

else if($ext=="c"){

 move_uploaded_file($_FILES["file"]["tmp_name"],$st1."/".$pid.".c");
```

```
$current = "C:\Dev-Cpp\bin\gcc users\\". $u."\\". $pid."\\".$tstamp."\\". $pid.".c -o users\\".
$u."\\". $pid."\\".$tstamp."\\". $u .".exe 2> users\\". $u."\\". $pid."\\".$tstamp."\\output.txt \n";}

else if($ext=="pas"){

 move_uploaded_file($_FILES["file"]["tmp_name"],$st1."/".$pid.".pas");

$current = "C:\\FPC\\2.6.2\\bin\\i386-win32\\fpc users\\". $u."\\". $pid."\\".$tstamp."\\".
$pid.".pas -o.\\users\\". $u."\\". $pid."\\".$tstamp."\\". $u .".exe > users\\". $u."\\".
$pid."\\".$tstamp."\\output.txt \n";}

file_put_contents($file, $current);

exec($file);
```

Οι εντολές γράφονται μέσα στο αρχείο «test.bat» το οποίο εκτελείται με την εντολή exec(). Αν υπάρχει πρόβλημα στο compile αυτό καταχωρείται στο αρχείο output.txt. Αν δεν υπάρχει πρόβλημα τότε προχωρούμε στο επόμενο βήμα.

2. Σε αυτό το βήμα δημιουργείται ένα batch file με το όνομα timeset το οποίο είναι υπεύθυνο να ελέγχει τον χρόνο εκτέλεσης του προγράμματος. Σε κανένα πρόγραμμα δεν επιτρέπετε να εκτελείται για περισσότερο από 2 δευτερόλεπτα. Ένα παράδειγμα από το timeset είναι το παρακάτω[07].

```
@echo off

for /F "tokens=1-4 delims=:.," %%a in ("%time%") do (

  set /A "start=(((%%a*60)+1%%b %% 100)*60+1%%c %% 100)*100+1%%d %% 100"

SET SS=%%c)

:loop

for /F "tokens=1-4 delims=:.," %%a in ("%time%") do (
```

```
 set /A "start=(((%%a*60)+1%%b %% 100)*60+1%%c %% 100)*100+1%%d %% 100"

SET SS1=%%c)

set /A elapsed=SS1-SS

if %elapsed% LSS 2 goto loop

taskkill /F /im pavlos.exe

exit
```

Μόλις ξεκινήσει η εκτέλεση των εντολών, χρόνος, σε δευτερόλεπτα αποθηκεύεται στη μεταβλητή SS. Η μεταβλητή SS1 αποθηκεύει τα τρέχων δευτερόλεπτα. Μόλις ο χρόνος που έχει περάσει (elapsed) φτάσει στο 2, σταματά η εκτέλεση του προγράμματος. Να σημειώσω ότι το αρχείο timeset δεν εκτελείται σε αυτή τη φάση. Απλά δημιουργείται.

3. Σε αυτή τη φάση το εκτελέσιμο που έχει δημιουργηθεί στο βήμα 1 θα δοκιμαστεί και θα παράξει outputs για τα inputs του προβλήματος. Το αποτέλεσμα για κάθε output που παράγεται αποθηκεύεται σε σχετικό αρχείο μέσα στο φάκελο υποβολής του προγράμματος. Για να γίνει αυτό γράφετε ξανά το test.bat. Μέσα σε αυτό τώρα θα καλέσω το timeset και θα τρέξω το εκτελέσιμο για συγκεκριμένο input. Όλη η διαδικασία βρίσκεται μέσα μια δομή επανάληψης και θα εκτελεστεί τόσες φορές, όσες είναι και ο αριθμός των test cases. [07]

```
start /min timeset.bat

users\pavlos\6\2014-01-07-06-29-36\pavlos.exe <askiseis\6\in5.txt > users\pavlos\6\2014-01-07-06-29-36\output5.txt
```

Οι πιο πάνω εντολές ξεκινούν ασύγχρονα το timeset.bat και δοκιμάζουν το pavlos.exe για το 5ο αρχείο εισόδου της άσκησης 6. Τα αποτελέσματα αποθηκεύονται στο αρχείο output5.txt.

4. Στη τελευταία φάση έχουμε τον έλεγχο/σύγκριση των outputs που έχουν παραχθεί από τον κώδικα που ανέβασε ο χρήστης με τα outputs που έχει ανεβάσει στο σύστημα ο συγγραφέας του προβλήματος. Αυτό γίνεται με τις πιο κάτω εντολές [07]

```
$f1=fopen("users\\". $u."\\". $pid."\\".$tstamp."\\output".$i.".txt","r");

$f2=fopen("askiseis\\".$pid."\\out".$i.".txt","r");

$ans1=fgets($f1);

 $ans2=fgets($f2);

if($ans1==$ans2){

            $color="90EE90";

            $msg="Σωστή Απάντηση";

            $points=$points+10;

          }

else{

            $color="FF4500";

            $msg="Λάθος Απάντηση";

          }
```

Οι σωστές απαντήσεις επιβραβεύονται με 10 βαθμούς και θα εμφανίζονται στο χρήστη με πράσινο ενώ οι λανθασμένες θα εμφανίζονται με πορτοκαλί..

5.5 Επιλογή ασκήσεων και test cases

Ένας από τους λόγους που αποφασίσαμε τη δημιουργία του συστήματος ήταν ότι θέλαμε να προσαρμόσουμε τις ασκήσεις στα μέτρα των δικών μας μαθητών με καλύτερη διαβάθμιση της δυσκολίας των ασκήσεων.

Η επιλογή μιας άσκησης για να εξετάσει μια κατηγορία αλγορίθμων είναι ένα θέμα, το πώς θα γίνει ο έλεγχος αυτής της άσκησης είναι ένα άλλο. Η επιλογή των test case που θα ελέγχει την ορθότητα του κώδικα που θα υποβάλλεται από τους μαθητές πρέπει να γίνεται με προσοχή και γενικά να ακολουθεί τους πιο κάτω κανόνες:

- Ελέγχει τις συνηθισμένες περιπτώσεις
- Ελέγχει τις ακραίες περιπτώσεις.
- Αναγνωρίζει σημεία στα οποία θα μπορούσε να κάνει λάθος ένας μαθητής και ελέγχει αυτά τα σημεία.
- Ελέγχει τα όρια των δεδομένων εισόδου

Οι ασκήσεις που ακολουθούν ανήκουν στις πιο κάτω κατηγορίες:

- Ακολουθιακή Δομή
- Δομή Διακλάδωσης
- Δομή Επανάληψης
- Πίνακες
- Στοίβες
- Γράφοι

Η κάθε άσκηση περιλαμβάνει την εκφώνηση, την επεξήγηση της, ενδεικτική λύση σε C++ και τα test cases.

Τέλος να αναφέρω κάποιους κανονισμούς/παραδοχές για τις ασκήσεις, που ισχύουν σχεδόν στην πλειοψηφία παρόμοιων συστημάτων.

- Για τις ασκήσεις χρησιμοποιούμε stin-stout και δεν υπάρχουν μηνύματα προς το χρήστη

- Τα όρια που δίνονται για κάθε άσκηση δε χρειάζεται να ελέγχονται.

- Η έξοδος πρέπει να εμφανίζεται με τον ίδιο τρόπο που δίνεται στο παράδειγμα της άσκησης.

- Στο τέλος πρέπει να έχουμε πάντα αλλαγή γραμμής

Σχετικές οδηγίες υπάρχουν και στην αρχική σελίδα του συστήματος.

5.5.1 Ακολουθιακή Δομή

Οι ασκήσεις αυτής της κατηγορίας είναι οι πιο εύκολες και στην πλειοψηφία των online συστημάτων που υπάρχουν σε άλλες χώρες παραλείπονται. Επειδή όμως οι μαθητές μας αργούν να έρθουν σε επαφή με προγραμματισμό θεωρώ ότι είναι αναγκαίες. Επίσης είναι σημαντικές ώστε οι μαθητές να εξοικειωθούν με το σύστημα.

Στις ασκήσεις ακολουθιακή δομής έχουμε είσοδο δεδομένων, επεξεργασία μέσα από κάποιους υπολογισμούς και παρουσίαση των αποτελεσμάτων.

Άσκηση 1

Εκφώνηση Να δημιουργήσετε το πρόγραμμα που δέχεται 2 ακέραιους αριθμούς και υπολογίζει και τυπώνει το άθροισμα. Παράδειγμα Εισόδου 2 3 Παράδειγμα Εξόδου 5

Επεξήγηση

Πρόκειται για μια πολύ απλή άσκηση που σκοπό έχει να ελέγξει ότι οι μαθητές μπορούν να διαβάσουν 2 αριθμούς, να τους προσθέσουν και να παρουσιάσουν το αποτέλεσμα.

Ενδεικτική Λύση

```
#include <iostream>
using namespace std;
int main(){
   int a,b,c;
   cin>>a>>b;
   c=a+b;
   cout<<c;
   return 0;
}
```

Test Cases

A/A	Input	Output
1	4 5	9
2	0 129	129
3	0 0	0
4	-1 2	1
5	-10894 5000	-5894

Άσκηση 2

Εκφώνηση

Η Αριάδνη έχει μάθει στο σχολείο ότι για να υπολογίσει την υποτείνουσα ενός ορθογωνίου, όταν είναι γνωστές οι 2 κάθετες πλευρές του, θα πρέπει να χρησιμοποιήσει το Πυθαγόρειο Θεώρημα.

Έχοντας αυτό υπόψη, η Αριάδνη σας καλεί να γράψετε ένα πρόγραμμα που θα δέχεται σαν είσοδο τις 2 κάθετες πλευρές του τριγώνου και θα υπολογίζει το εμβαδόν και την περίμετρο του.

Σημείωση: Όλοι οι αριθμοί είναι δεκαδικοί και το αποτέλεσμα εμφανίζεται με την ακρίβεια 2 δεκαδικών ψηφίων

Παράδειγμα Εισόδου

3 4

Παράδειγμα Εξόδου

6.00

12.00

Επεξήγηση

Σε αυτή την άσκηση ελέγχουμε ότι ο μαθητής μπορεί να αντιληφθεί την αναγκαιότητα των ενδιάμεσων υπολογισμών (υποτείνουσα). Επίσης θέλουμε να τον αναγκάσουμε να χρησιμοποιήσεις έτοιμες συναρτήσεις (τετραγωνική ρίζα) και να ελέγξει την ακρίβεια των δεκαδικών ψηφίων στο αποτέλεσμα.

Ενδεικτική Λύση

```
#include <iostream>
#include <iomanip>
#include <cmath>
using namespace std;
int main(){
   float a,b,c,e,p;
   cin>>a>>b;
```

```
  c=sqrt(a*a+b*b);
  e=a*b/2;
  p=a+b+c;
  cout<<fixed<<setprecision(2)<<e<<endl;
  cout<<fixed<<setprecision(2)<<p<<endl;
  return 0;
}
```

Test Cases

A/A	Input	Output
1	3 4	6.00 12.00
2	6 1	18.00 20.49
3	1 1	0.50 3.41
4	100 150	7500.00 430.28
5	4500 1586	3568500.00 10857.31

Άσκηση 3

Εκφώνηση

Να δημιουργήσετε το πρόγραμμα που δέχεται έναν ακέραιο τριψήφιο αριθμό και υπολογίζει το τετράγωνο του αθροίσματος των ψηφίων του.

Παράδειγμα Εισόδου

123

Παράδειγμα Εξόδου

36

Επεξήγηση

Η άσκηση αυτή έχει ως σκοπό την εισαγωγή των μαθητών στην έννοια του υπολοίπου (modulo).

Ενδεικτική Λύση

```
#include <iostream>
using namespace std;
int main(){
  int a,m,d,e,result;

  cin>>a;
  e=a/100;
  d=a%100/10;
  m=a%10;
  result=(e+d+m)*(e+d+m);
  cout<<result<<endl;
  return 0;
}
```

Test Cases

Στο test 2 ελέγχουμε την περίπτωση οι δεκάδες να είναι 0, στο 3 οι μονάδες να είναι 0, στο 4 να είναι 0 τόσο οι μονάδες όσο και οι εκατοντάδες.

A/A	Input	Output
1	123	36
2	209	121

3	310	16
4	400	16
5	555	225

Άσκηση 4

Εκφώνηση

Η Αριάδνη θέλει να υπολογίσει το γενικό της σε ένα μη εξεταζόμενο μάθημα. Ο γενικός βαθμός υπολογίζεται ως εξής: (βαθμός α τετραμήνου +βαθμός β τετραμήνου)/2.

Οι βαθμοί είναι ακέραιοι αριθμοί από το 1 μέχρι το 20, ενώ ο γενικός βαθμός δεκαδικός και παρουσιάζεται με ακρίβεια 2 δεκαδικών ψηφίων.

Παράδειγμα Εισόδου

15 16

Παράδειγμα Εξόδου

15.50

Επεξήγηση

Η άσκηση αυτή έχει ως σκοπό να αντιληφθούν οι μαθητές ότι η διαίρεση μεταξύ ακεραίων αριθμών δε δίνει σαν αποτέλεσμα έναν δεκαδικό. Θα πρέπει να γίνει ένας από τους ακεραίους cast σε δεκαδικό.

Ενδεικτική Λύση

```
#include <iostream>
```

```
#include <iomanip>
using namespace std;
int main(){
  int a,b;
  float mo;
  cin>>a>>b;
  mo=(float)(a+b)/2;
  cout<<fixed<<setprecision(2)<<mo<<endl;
  return 0;
}
```

Test Cases

A/A	Input	Output
1	15 16	15.50
2	20 20	20.00
3	1 20	10.50
4	10 19	14.50
5	2 3	2.50

5.5.2 Δομή Διακλάδωσης

Στις ασκήσεις της δομής διακλάδωσης, οι μαθητές καλούνται να δημιουργήσουν προγράμματα τα οποία θα πρέπει να παίρνουν διάφορες αποφάσεις. Σε αυτή τη κατηγορία οι πιο απλές ασκήσεις έχουν μια μόνο διακλάδωση. Μπορούμε επίσης να έχουμε ασκήσεις με σύνθετες συνθήκες χρησιμοποιώντας λογικούς τελεστές καθώς και προβλήματα με φωλιασμένες δομές (nested if).

Άσκηση 1

Εκφώνηση

Ο τελικός βαθμός ενός μαθητή σε ένα μάθημα υπολογίζεται με βάση την προφορική και τη γραπτή του βαθμολογία με την ακόλουθη διαδικασία:

Αν η διαφορά των δύο βαθμών είναι μεγαλύτερη από πέντε (5) μονάδες, τότε ο προφορικός βαθμός προσαρμόζεται (δηλαδή αυξάνεται ή μειώνεται) έτσι ώστε η αντίστοιχη διαφορά να μειωθεί στις τρεις (3) μονάδες, διαφορετικά ο προφορικός βαθμός παραμένει αμετάβλητος. Ο τελικός βαθμός είναι ο μέσος όρος του προφορικού και του γραπτού βαθμού.

Παράδειγμα προσαρμογής του προφορικού βαθμού:

Αν ο γραπτός βαθμός είναι 18 και ο προφορικός 11, τότε ο προφορικός γίνεται 15, ενώ αν ο γραπτός βαθμός είναι 10 και ο προφορικός 19, τότε ο προφορικός γίνεται 13.

Είσοδος

Δύο ακέραιο αριθμοί Χ,Υ (0<=Χ,Υ<=20) που αντιστοιχούν στο γραπτό και τον προφορικό βαθμό.

Έξοδος

Η τελική βαθμολογία είναι ένας δεκαδικός αριθμός με ακρίβεια 2 δεκαδικών ψηφίων.

Παράδειγμα Εισόδου 1

18 11

Παράδειγμα Εξόδου 1

16.50

Παράδειγμα Εισόδου 2

10 12

Παράδειγμα Εξόδου 2

11.00

Επεξήγηση

Πρόκειται για μια απλή άσκηση η οποία μπορεί να λυθεί ελέγχοντας 2 συνθήκες. Η 1η συνθήκη ελέγχει ότι υπάρχει πράγματι διαφορά 5 μονάδων και η δεύτερη ελέγχει ποιος βαθμός είναι ο μεγαλύτερος

Ενδεικτική Λύση

```
#include<iostream>
#include <cmath>
#include <iomanip>
using namespace std;
int main(){
 int a,b;
 float mo;
   cin>>a>>b;
   if(abs(a-b)>5)
    if(a>b)
      b=a-3;
    else
      b=a+3;
   mo=(a+b)/2.0;
   cout<<fixed<<setprecision(2)<<mo;
 return 0;
}
```

Test Cases

Στο 1ο , 4ο δεν υπάρχει προσαρμογή, στο 2ο υπάρχει προσαρμογή με τον προφορικό να είναι μεγαλύτερος, στο 3ο υπάρχει προσαρμογή με το γραπτό να είναι μεγαλύτερος και στο 5ο ελέγχεται ότι δεν έχει μπει η ισότητα στη σύγκριση.

A/A	Input	Output
1	18 19	18.50
2	11 17	12.50
3	20 2	18.50
4	18 18	18.00
5	12 17	14.50

Άσκηση 2

Εκφώνηση

Να γράψετε το πρόγραμμα που διαβάζει 3 ακέραιους αριθμούς και υπολογίζει και τυπώνει το μεσαίο σε μέγεθος.

Παράδειγμα εισόδου

30 107 54

Παράδειγμα εξόδου

54

Επεξήγηση

Σε αυτήν την απλή άσκηση εξετάζουμε τις σύνθετες συνθήκες και την φωλιασμένη δομή διακλάδωσης.

Ενδεικτική Λύση

```
#include<iostream>
using namespace std;
int main(){
 int a,b,c;
 cin>>a>>b>>c;
 if((a>b)&&(a<c)||(a<b)&&(a>c))
  cout<<a<<endl;
 else if ((b>a)&&(b<c)||(b<a)&&(b>c))
  cout<<b<<endl;
 else
  cout<<c <<endl;
 return 0;
}
```

Test Cases

A/A	Input	Output
1	10 20 30	18.50
2	7 30 1	7
3	-1 1 0	0
4	-1 -3 -2	-2
5	11 21 31	21

Άσκηση 3

Εκφώνηση

Μια ομάδα εισβολέων συστημάτων με την επωνυμία Eponymous έχουν κοινοποιήσει ένα βίντεο στα μέσα μαζικής ενημέρωσης. Το βίντεο ενημερώνει τους αρμόδιους φορείς ότι την 12η Δεκεμβρίου του 2012 η ώρα 12:12 μμ θα ξεκινήσει μια αντίστροφη μέτρηση η οποία θα διαρκέσει 12 ώρες ακριβώς. Αν μέσα σε αυτές τις 12 ώρες δεν ικανοποιηθούν όλα τα αιτήματά

τους, μεταξύ άλλων και η δωρεάν πρόσβαση στο διαδίκτυο για το κοινό, θα προσβάλουν με ένα επικίνδυνο ιό επονομαζόμενο Flame όλες τις κυβερνητικές υπηρεσίες.

Η Υπηρεσία Πληροφοριών έχει αναθέσει στην Αριάδνη τον εντοπισμό και την εξουδετέρωση του Flame. Η Αριάδνη σαν έμπειρη προγραμματίστρια, ειδική στην αντιμετώπιση κρίσεων φέρνει σε πέρας την αποστολή και εντοπίζει τον ιό. Είναι όμως αρκετά γρήγορη; Έχει προλάβει το χρονικό περιθώριο των 12 ωρών;

Είσοδος

Τρεις ακέραιοι, D (12<=D<=31), H (0<=H<=23) και M (0<=59) που υποδηλώνουν την ημερομηνία και την ώρα που εξουδετερώθηκε ο ιός. Για παράδειγμα 12 23 34 σημαίνει ότι ο ιός εξουδετερώθηκε την 12η Δεκεμβρίου η ώρα 11:34 μμ.

Έξοδος

Ο συνολικός αριθμός των λεπτών που έχουν περάσει από τις 12 Δεκεμβρίου 2012 η ώρα 12:12 μμ και το μήνυμα «You made it!» αν δεν έχουν περάσει 12 ώρες ή «Too late» σε αντίθετη περίπτωση. Το πρόγραμμα να επιστρέφει -1 αν υπάρχει λάθος στα δεδομένα εισόδου.

Παράδειγμα εισόδου 1

12 23 34

Παράδειγμα εξόδου 1

682

You made it!

Παράδειγμα Εισόδου 2

13 12 11

Παράδειγμα Εξόδου 2

1439

Too late

Παράδειγμα Εισόδου 3

0 12 11

Παράδειγμα Εξόδου 3

-1

Επεξήγηση

Η άσκηση αυτή εξετάζει και πάλι τις σύνθετες συνθήκες και τις φωλιασμένες δομές με έναν πιο πολύπλοκο τρόπο. Πρέπει να γίνονται δύο διαφορετικοί έλεγχοι, για την εγκυρότητα των στοιχείων που εισάγονται και για τον αν έχει προλάβει ή όχι η Αριάδνη.

Ενδεικτική Λύση

```
#include<iostream>
using namespace std;
int main(){
  int d, h, m,a;

  cin>>d>>h>>m;
  if((d<12)||((d==12)&&(h<12))||((d==12)&&(h==12)&&(m<12))){
        cout<<-1<<"\n";
  }
  else{
   if(d==12){
     if(h==12){
```

```
        a=m-12;
        cout<<m-12<<"\n";
      }
   else{
    a=(h-13)*60+(60-12)+m;
    cout<<(h-13)*60+(60-12)+m<<"\n";
   }
   }
   else{
      a=708+(d-13)*60+h*60+m;
      cout<<708+(d-13)*60+h*60+m<<"\n";
   }
   if(a<(12*60)){
      cout<<"You made it!\n";
   }
   else{
      cout<<"Too late\n";
   }
   }
      return 0;
}
```

Test Cases

Τα τεστ 1 και 5 ελέγχουν την ορθότητα των στοιχείων, το 2ο τεστ ελέγχει ότι έχοντας περάσει στην επόμενη μέρα είναι ακόμα μέσα στα όρια, ενώ το 4ο ελέγχει ότι είναι μέσα στα όρια ενώ βρίσκεται μέσα στην ίδια μέρα. Τέλος το 3ο τεστ ελέγχει την περίπτωση που έχουμε υπερβεί το όριο.

A/A	Input	Output
1	10 40 55	-1
2	13 0 0	708 You made it!

3	13 12 12	1440 Too late
4	12 12 14	2 You made it!
5	10 400 55	-1

Άσκηση 4

Εκφώνηση

Το σχολείο της Αριάδνης ετοιμάζεται να επισκεφθεί τον αρχαιολογικό χώρο της Χοιροκοιτίας. Η Αριάδνη είναι υπεύθυνη να πάρει τηλέφωνο και να ρωτήσει για την έκπτωση που έχει το σχολείο. Ο υπεύθυνος του αρχαιολογικού χώρου ενημέρωσε την Αριάδνη ότι αν πάνε τουλάχιστον Α άτομα το σχολείο θα έχει 10% έκπτωση, αν πάνε Β άτομα θα έχει 20% και αν πάνε Γ άτομα θα έχει 30%. Η κανονική τιμή του εισιτηρίου είναι 10 ευρώ.

Ο διευθυντής του σχολείου ενημέρωσε την Αριάδνη ότι θα πάνε στην εκδρομή Ν άτομα. Βοηθήστε την Αριάδνη να φτιάξει ένα πρόγραμμα που θα υπολογίζει το ποσό που πρέπει να πληρώσει το σχολείο. Προσοχή σε ορισμένες περιπτώσεις μπορεί να συμφέρει να αγοραστούν περισσότερα εισιτήρια.

Δεδομένα Εισόδου

Η πρώτη γραμμή θα περιέχει έναν ακέραιο αριθμό Ν (Ν <= 10000). Η δεύτερη γραμμή θα περιέχει τους 3 ακεραίους Α, Β, και Γ (Α,Β,Γ <= 10000)

Δεδομένα Εξόδου

Ένας ακέραιος που θα παρουσιάζει την ελάχιστη συνολική τιμή.

Παράδειγμα Εισόδου 1

30

10 20 50

Παράδειγμα Εξόδου 1

240

Παράδειγμα Εισόδου 2

48

10 20 50

Παράδειγμα Εξόδου 2

336

Επεξήγηση

Σε αυτήν την απλή άσκηση εξετάζουμε τις σύνθετες συνθήκες και την φωλιασμένη δομή διακλάδωσης.

Ενδεικτική Λύση

```
#include<iostream>
using namespace std;
int main(){
  int N,A,B,C,total,max;

 cin>>N>>A>>B>>C;
  if(N<A)
    total=N*10;
  else
  if(N<B)
    total=N*10*90/100;
```

```
  else
  if(N<C)
    total=N*10*80/100;
  else
    total=N*10*70/100;
  if((N<A)&&(total>(A*10*90/100)))
    total=A*10*90/100;
  if((N<B)&&(total>(B*10*80/100)))
    total=B*10*80/100;
  if((N<C)&&(total>(C*10*70/100)))
    total=C*10*70/100;
  cout<<total;
return 0;}
```

Test Cases

Τα τεστ 1 μέχρι 4 ελέγχουν τα όρια των αριθμών Α, Β, Γ. Τα τεστ 5 με 7 ελέγχουν την περίπτωση να βρίσκεται σε μια κατηγορία και να μας συμφέρει να πάμε στην επόμενη κατηγορία. Το τεστ 8 ελέγχει την περίπτωση να μας συμφέρει να περάσουμε σε ακόμα πιο μεγάλη κατηγορία.

A/A	Input	Output
1	2 10 20 30	20
2	13 8 20 50	117
3	21 10 21 30	168
4	50 30 35 40	350
5	9 10 20 30	90
6	195 100 200 300	1600

7	2997 1000 2000 3000	21000
8	19 20 21 22	154

5.5.3 Δομή Επανάληψης

Στη δομή επανάληψης οι μαθητές πρέπει να εξοικειωθούν με την ιδέα των επαναλαμβανόμενων εντολών. Η εμπειρία μου, μου έχει δείξει ότι με τέτοιου είδους ασκήσεις μπερδεύονται πολύ στην αρχή. Για το λόγο αυτό οι 2 πρώτες ασκήσεις θα είναι αρκετά εύκολες.

Άσκηση 1

Εκφώνηση

Να δημιουργήσετε το πρόγραμμα που δέχεται 2 ακέραιους θετικούς αριθμούς και υπολογίζει και τυπώνει το μέγιστο κοινό διαιρέτη.

Παράδειγμα Εισόδου

12 8

Παράδειγμα Εξόδου

4

Επεξήγηση

Σε αυτήν την απλή άσκηση εξετάζουμε την απλή δομή επανάληψης με ένα έλεγχο μέσα στην επανάληψη.

Ενδεικτική Λύση

```
#include<iostream>
using namespace std;
int main(){
 int a,b,i,mcd;
   cin>>a>>b;
   for(i=min(a,b);i>=1;i--){
     if ((a%i==0)&&(b%i==0)){
        mcd=i;
        break;
     }
   }
      cout<<mcd<<endl;
  return 0;
}
```

Test Cases

A/A	Input	Output
1	12 6	6
2	8 16	8
3	20 20	20
4	3 7	1
5	12 8	4

Άσκηση 2

Εκφώνηση

Η Αριάδνη και ο Κυριάκος παίζουν ένα παιχνίδι. Η Αριάδνη λέει στον Κυριάκο 10 θετικούς ακέραιους αριθμούς και ο Κυριάκος πρέπει να απαντήσει με το άθροισμα των πρώτων αριθμών.

Είσοδος

Δέκα ακέραιοι θετικοί αριθμοί

Έξοδος

Το άθροισμα των πρώτων αριθμών.

Ένας αριθμός ονομάζεται πρώτος όταν διαιρείται μόνο με το 1 και τον εαυτό του. Ο αριθμός 1 ΔΕΝ είναι πρώτος.

Παράδειγμα Εισόδου
7
9
21
3
1
2
17
30
46
88
Παράδειγμα Εξόδου
29

Επεξήγηση

Στην άσκηση αυτή εξετάζουμε την δυνατότητα των μαθητών να τοποθετήσουν μια δομή επανάληψης μέσα σε μια άλλη

Ενδεικτική Λύση

```
#include<iostream>
#include<cmath>
using namespace std;
int main(){
 int i,sum=0,n,j,flag;
  for(i=1;i<=10;i++){
    cin>>n;
    flag=0;
    for(j=2;j<=sqrt(n);j++)
       if ((n%j)==0){
          flag=1;
         break;
       }
    if((flag==0)&&(n!=1))
    {
       sum=sum+n;
    }
  }
  cout<<sum<<endl;
 return 0;
}
```

Test Cases

Το 2ο τεστ περιέχει μόνο πρώτους, ενώ το 3ο δεν περιέχει κανένα.

A/A	Input	Output
1	7 9 21 3 1 2 17 30 46 88	29
2	2 2 2 2 2 2 2 2 2 2	20
3	4 4 4 4 4 4 4 4 4 4	0
4	1 2 3 4 5 6 7 8 9 10	17
5	100 5003 97 503 66 90 10001 76 8 2222	5603

Άσκηση 3

Εκφώνηση

Με ψηφία από το σύνολο {4,5,6,7,8} σχηματίζουμε τριψήφιους φυσικούς αριθμούς χωρίς επανάληψη ψηφίου. Για παράδειγμα δύο έγκυροι, τριψήφιοι από το πιο πάνω σύνολο είναι ο αριθμός 456 και ο αριθμός 876.

Να βρείτε:

α) το πλήθος των τριψήφιων φυσικών αριθμών που σχηματίζονται και

β) το άθροισμα όλων των τριψήφιων φυσικών αριθμών που σχηματίζονται

Από τις Παγκύπριες εξετάσεις Μαθηματικών 2012. Μέρος Α', Άσκηση 10.

1. Να γράψετε ένα πρόγραμμα που να δέχεται 5 συνεχόμενα ψηφία από το 0 μέχρι το 9.

2. Να γίνεται έλεγχος για την εγκυρότητα των ψηφίων και να ενημερώνεται ο χρήστης για την ορθότητά τους. Αν έχει γίνει λάθος στην εισαγωγή των ψηφίων να δίνεται η δυνατότητα στο χρήστη να επαναλάβει τη διαδικασία.

Για παράδειγμα, αν ο χρήστης δώσει τους αριθμούς 0, 1, 2, 3, 4 τότε τα ψηφία είναι έγκυρα και θα εμφανιστούν τα αποτελέσματα. Αν ο χρήστης δώσει τα ψηφία 1, 3, 5, 7, 9 τα ψηφία δεν είναι συνεχόμενα και επομένως δεν είναι έγκυρα. Σε αυτή την περίπτωση το πρόγραμμα θα εμφανίζει το μήνυμα «Wrong Input. Please try again.» και θα επιτρέπει στο χρήστη να ξαναδώσει αριθμούς.

3. Όταν δοθούν οι έγκυροι αριθμοί 0, 1, 2, 3, 4 το πρόγραμμα σας θα πρέπει να εμφανίζει τα ζητούμενα όπως πιο κάτω:

48

12990

Παράδειγμα εισόδου

1 3 5 7 9

1 2 4 5 6

1 2 3 4 10

11 12 13 14 15

0 0 1 2 2

0 1 2 3 -4

0 0 0 0 0

9 8 7 6 5

-5 -4 -3 -2 -1

4 5 6 7 8

Παράδειγμα εξόδου

Wrong Input. Please try again.

Wrong Input. Please try again.

Wrong Input. Please try again.

Wrong Input. Please try again.

Wrong Input. Please try again.

Wrong Input. Please try again.

Wrong Input. Please try again.

Wrong Input. Please try again.

Wrong Input. Please try again.

60

39960

Επεξήγηση

Σε αυτή την άσκηση ο μαθητής καλείται να χρησιμοποιήσει πολλαπλές δομές επανάληψης. Θα έχει μια επανάληψη για να ελέγχει την ορθότητα και μια φωλιασμένη για τους υπολογισμούς.

Ενδεικτική Λύση

```
#include <iostream>
using namespace std;

int main() {
int n1, n2, n3, n4, n5;
int d1,d2,d3,num;
int count=0;
int sum=0;

cin >> n1 >> n2 >> n3 >> n4 >> n5;
while ((n1!=n2-1) || (n2!=n3-1) || (n3!=n4-1) || (n4!=n5-1) || (n1<0)|| (n5>9)) {
   cout << "Wrong Input. Please try again." <<endl;
   cin >> n1 >> n2 >> n3 >> n4 >> n5;
 }

 for (d1=n1; d1<=n5; d1++)
  for (d2=n1; d2<=n5; d2++)
   for (d3=n1; d3<=n5; d3++)
   if ((d1!=d2) && (d1!=0) && (d1!=d3) && (d2!=d3)) {
    count++;
    num=d1*100+d2*10+d3;
    sum=sum+num;
   }
   cout<<count<<endl;
   cout<<sum<<endl;
   return 0;}
```

Test Cases

Στο 1ο τεστ έχουμε σωστή σειρά με τον αριθμό μηδέν να περιλαμβάνεται μέσα στους αριθμούς. Στα τεστ 2 και 5 έχουμε σωστές εισόδους, χωρίς το 0. Στο τεστ 3 έχουμε μια λανθασμένη είσοδο, ενώ στο τεστ 4 δύο.

A/A	Input	Output
1	0 1 2 3 4	48 12990
2	1 2 3 4 5	60 19980
3	7 8 9 10 11 2 3 4 5 6	Wrong Input. Please try again. 60 26640
4	-5 -4 -3 -2 -1 1 2 3 4 6 5 6 7 8 9	Wrong Input. Please try again. Wrong Input. Please try again. 60 46620
5	4 5 6 7 8	60 39960

Άσκηση 4

Εκφώνηση

Ο θυρωρός ενός ουρανοξύστη θέλει να γνωρίζει σε ποιο όροφο έχει σταματήσει ο ανελκυστήρας στο τέλος μιας ημέρας καθώς επίσης και την απόσταση (σε ορόφους) που έχει διανύσει. (κάθε όροφος θεωρείται ότι έχει ύψος 3 μέτρα) Ο ανελκυστήρας ξεκινά καθημερινά από το ισόγειο (όροφος 0) και μετακινείται είτε προς τα πάνω, είτε προς τα κάτω. Ο ουρανοξύστης έχει 75 ορόφους και 5 υπόγειους ορόφους που χρησιμεύουν σαν χώροι στάθμευσης (σύνολο 80 ορόφους).

Δίνονται σαν δεδομένα εισόδου οι μετακινήσεις του ανελκυστήρα με τη μορφή θετικών ή αρνητικών ακεραίων που υποδεικνύουν μετακίνηση ορόφων προς τα πάνω ή προς τα κάτω

αντίστοιχα. Όταν δοθεί ο αριθμός μηδέν σημαίνει ότι ο ανελκυστήρας έχει σταματήσει τη λειτουργία του. Να γράψετε το πρόγραμμα που δέχεται τις μετακινήσεις του ανελκυστήρα (μέχρι να δοθεί ο αριθμός μηδέν) και επιστρέφει τον όροφο που έχει σταματήσει και την απόσταση που έχει διανύσει σε μέτρα.

Παράδειγμα Εισόδου

23
-5
31
25
-40
23
5
-10
0
Όροφος: 52
Απόσταση: 162 *3 = 486 μέτρα

Επεξήγηση

Στις προηγούμενες ασκήσεις ο μαθητής ήξερε τον αριθμό των επαναλήψεων. Σε αυτή την άσκηση είναι αναγκασμένος να χρησιμοποιήσει την εντολή while.

Ενδεικτική Λύση

```
#include<iostream>
#include<cmath>
using namespace std;
int main(){
 int m=0,n,orofos=0;
   cin>>n;
```

```
  while (n!=0){
    orofos=orofos+n;
    m=m+abs(n);
    cin>>n;
  }
  cout<<orofos<<endl;
  cout<<m*3<<endl;
 return 0;
}
```

Test Cases

Στο 1ο τεστ ανελκυστήρας κινείται μόνο προς τα κάτω. Στο 2ο δε μετακινείται καθόλου. Στο 4ο σταματά στον ψηλότερο όροφο.

A/A	Input	Output
1	-1 -1 0	-2 6
2	0	0 0
3	1 -1 2 -2 3 -3 1 0	1 39
4	75 -75 75 0	75 675
5	40 22 -45 5 -5 4 3 -7 0	17 393

5.5.4 Πίνακες

Οι πίνακες είναι η απλούστερη δομή δεδομένων που πρέπει να μάθουν να χρησιμοποιούν οι μαθητές. Είναι εύκολος στη δημιουργία του και στην προσπέλαση του. Για τους πίνακες υπάρχει μια μεγάλη γκάμα ασκήσεων από τις οποίες μπορούμε να επιλέξουμε. Λόγω περιορισμένου χώρου και χρόνου έχω επιλέξει 4 ασκήσεις που πιστεύω ότι αντιπροσωπεύουν μεγάλες

κατηγορίες ασκήσεων. Η 1η άσκηση χρησιμοποιεί μονοδιάστατους πίνακες, η 2η πίνακες χαρακτήρων (strings), η 3η δισδιάστατους πίνακες και η 4η χρησιμοποιεί ταξινόμηση.

Άσκηση 1

Εκφώνηση

Η Αριάδνη θέλει να επισκεφθεί τον Κυριάκο, τον παλιό συμμαθητή του, για πρώτη φορά μετά από πολλά χρόνια. Θέλοντας να του κάνει έκπληξη δεν έχει επικοινωνήσει προηγουμένως μαζί του τηλεφωνικά και αντιμετωπίζει τώρα ένα πρόβλημα. Ενώ θυμάται την οδό που μένει ο Κυριάκος δεν θυμάται τον αριθμό του σπιτιού του. Θυμάται όμως ότι ο αριθμός του σπιτιού του Κυριάκου είναι διπλάσιος από κάποιο άλλο σπίτι της ιδίας οδού και ταυτόχρονα ο μισός κάποιου άλλου σπιτιού της ιδίας οδού. Αν σας δίνεται ο συνολικός αριθμός των σπιτιών μαζί με τον αριθμό του κάθε σπιτιού να βοηθήσετε την Αριάδνη να βρει το σπίτι του Κυριάκου.

Δεδομένα εισόδου

Ένας ακέραιος αριθμός N (3<=N<=50), ο συνολικός αριθμός των σπιτιών στην οδό του Κυριάκου.

N ακέραιοι αριθμοί K (1<=K<=250), ο αριθμός του κάθε σπιτιού. Δύο σπίτια δεν μπορούν να έχουν τον ίδιο αριθμό.

Δεδομένα εξόδου

Ο αριθμός του σπιτιού του Κυριάκου.

Παράδειγμα εισόδου

5

1 26 13 52 32

Παράδειγμα Εξόδου

26

Επεξήγηση

Η άσκηση αποθηκεύει τους αριθμούς των σπιτιών σε ένα πίνακα και στη συνέχεια χρησιμοποιώντας brute force ελέγχει όλους τους αριθμούς αν έχουν διπλάσιο και μισό. Η άσκηση λύνεται και με πολυπλοκότητα $O(n^2)$

Ενδεικτική Λύση

```
#include <iostream>
using namespace std;
int main(){
int n;
cin >> n;
int nums[n];
for (int i=0; i<n; i++)
 cin >> nums[i];

for (int a=0; a<n; a++)
   for (int n1=0; n1<n;n1++)
     for (int n2=0; n2<n; n2++)
       if ((nums[a]==nums[n1]*2) && (nums[a]*2==nums[n2]))
         cout << nums[a]<< endl;
return 0;
}
```

Test Cases

Η δυσκολία που είχα στη δημιουργία αυτών των τεστ, ήταν ότι έπρεπε να βεβαιωθώ ότι είχαν μόνο μια πιθανή λύση.

A/A	Input	Output

1	7 3 8 24 48 51 12 99	24
2	4 40 30 20 10	20
3	10 1 3 5 7 13 17 21 23 34 68	34
4	20 2 3 5 7 11 13 17 19 21 29 31 37 42 43 47 53 59 61 67 84	42
5	50 200 173 179 181 191 193 197 199 211 223 227 229 2 3 5 7 11 13 103 107 109 113 127 131 137 139 149 151 157 163 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 50 100	100

Άσκηση 2

Εκφώνηση

Παλίνδρομη καλείται μια λέξη της οποίας οι χαρακτήρες της αν διαβαστούν από τον τελευταίο χαρακτήρα μέχρι τον πρώτο προκύπτει πάλι η ίδια λέξη, όπως για παράδειγμα οι λέξεις «level», «madam» ή «deified». Να γράψετε π ρόγραμμα το οποίο να διαβάζει έναν ακέραιο αριθμό Χ (0<Χ<=1000) που αντιστοιχεί στον αριθμό των συμβολοσειρών που ακολουθούν και να εμφανίζει το πλήθος των παλίνδρομων που βρέθηκαν.

Δεδομένα εισόδου

Ένας ακέραιος αριθμός Χ (0<Χ<=1000) ο αριθμός των συμβολοσειρών.

Στις επόμενες Χ γραμμές ακολουθούν συμβολοσειρές μεγέθους μέχρι 1000 χαρακτήρων.

Δεδομένα εξόδου

Ένας ακέραιος, το πλήθος των παλίνδρομων.

Παράδειγμα εισόδου

3

ANNA

@#$#@

abcd

Παράδειγμα εξόδου

2

Επεξήγηση

Η άσκηση ασχολείται με πίνακες χαρακτήρων (strings). Το πρόγραμμα πρέπει να διαβάσει τις Ν συμβολοσειρές και να αποφασίσει αν κάθε μια αποτελεί παλίνδρομο. Στο τέλος τυπώνεται το σύνολο των συμβολοσειρών που αποτελούν παλίνδρομα. Οι περιπτώσεις που θα εξετάσουμε είναι: α) να είναι/να μην είναι παλίνδρομο περιττού μεγέθους χαρακτήρων, β) να είναι/να μην είναι παλίνδρομο άρτιου μεγέθους χαρακτήρων και γ) να είναι μόνο ένας χαρακτήρας.

Ενδεικτική Λύση

```
#include <string>
#include <iostream>
using namespace std;
int main() {
   int N,size,i,p=0;
   string x;
   cin>>N;
   for(i=1;i<=N;i++){
      cin>>x;
      size=x.length();
      if (size==1){
         p++;
      }
      else {
      int flag=0;
      int j=0,k=size-1;
```

```
    while((flag==0)&&(j<k)){
      if(x[j]!=x[k])
        flag=1;
      j++;
      k--;
    }
    if(flag==0)
     p++;
    }
  }
  cout<<p<<endl;
  return 0;
}
```

<u>Test Cases</u>

<u>A/A</u>	<u>Input</u>	<u>Output</u>
1	3 asnna a abaaaa	1
2	5 anna annna a abab baba	3
3	11 aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa	11

	aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa	
4	1 qw	0
5	4 12345 12121212121212121212121212 fof w	2

Άσκηση 3

<u>Εκφώνηση</u> Να γραφεί πρόγραμμα που να δέχεται ως είσοδο έναν ορθογώνιο πίνακα από ακέραιους αριθμούς Α διαστάσεων NxM και τυπώνει τον ανάστροφο πίνακα Τ μετατρέποντας δηλαδή, τις

γραμμές σε στήλες και τις στήλες σε γραμμές. Για κάθε στοιχείο του πίνακα A και του ανάστροφου πίνακα T ισχύει A[i,j] = T[j,i].

Δεδομένα Εισόδου

Η πρώτη γραμμή εισόδου αποτελείται από δύο θετικούς ακέραιους αριθμούς N και M που αντιπροσωπεύουν τις διαστάσεις του πίνακα (όπου, N <= 40 και M <= 40). Οι επόμενες N γραμμές αποτελούνται από M θετικούς ακέραιους αριθμούς η κάθε μια, οι οποίοι διαχωρίζονται με κενό διάστημα και έχουν τιμές μικρότερες του 10. Αυτά είναι τα στοιχεία του πίνακα εισόδου.

Δεδομένα Εξόδου

Το πρόγραμμα επιστρέφει τις M γραμμές του ανάστροφου πίνακα με N στοιχεία η κάθε μια.

Παράδειγμα Εισόδου

2 3
1 2 3
4 5 6
Παράδειγμα Εξόδου
1 4
2 5
3 6

Επεξήγηση

Σε αυτή την άσκηση οι μαθητές καλούνται να χρησιμοποιήσουν πίνακες 2 διαστάσεων και να κατανοήσουν τις έννοιες των γραμμών και των στηλών

Ενδεικτική Λύση

```
#include <iostream>
#define N 40
#define M 40
using namespace std;
```

```
int main() {
  int a[N][M],i,j,r,c;
  int t[N][M];
  cin>>r>>c;
  for(i=0;i<r;i++)
    for(j=0;j<c;j++)
      cin>>a[i][j];
  for(i=0;i<r;i++){
    for(j=0;j<c;j++){
      t[j][i]=a[i][j];
    }
  }
  for(i=0;i<c;i++){
    for(j=0;j<r;j++){
      cout<<t[i][j]<<" ";
    }
    cout<<endl;
  }
  return 0;
}
```

Test Cases

Στο τεστ 2 γίνεται έλεγχος για τετραγωνικό πίνακα, στο τεστ 3 για πίνακα γραμμή, στο τεστ 4 για πίνακα στήλη και στο τεστ 5 για πίνακα με μόνο ένα στοιχείο.

A/A	Input	Output
1	2 3 1 2 3 4 5 6	1 4 2 5 3 6
2	3 3 1 2 3 4 5 6	1 4 7 2 5 8 3 6 9

	7 8 9	
3	1 10 1 2 3 4 5 6 7 8 9 10	1 2 3 4 5 6 7 8 9 10
4	10 1 1 2 3 4 5 6 7 8 9 10	1 2 3 4 5 6 7 8 9 10
5	1 1 40	40

Άσκηση 4

Εκφώνηση

Τρεις αγρότες σηκώνονται κάθε πρωί στις 5:00 και πηγαίνουν στο στάβλο για να αρμέξουν τρεις αγελάδες. Ο 1ος αγρότης ξεκινά να αρμέγει την αγελάδα του στην ώρα 300 (σημαίνει 300 δευτερόλεπτα μετά τις 5:00) και τελειώνει την ώρα 1000. Ο 2ος ξεκινά στην ώρα 700 και τελειώνει την ώρα 1200 ενώ ο 3ος στην ώρα 1500 και 2100 αντίστοιχα. Ο μεγαλύτερος χρόνος συνεχούς αρμέγματος (δηλαδή υπήρχε τουλάχιστον ένας αγρότης που άρμεγε) ήταν 900

δευτερόλεπτα (300 με 1200). Αντιστοίχως ο μέγιστος χρόνος όπου καμιά αγελάδα δεν αρμεγόταν ήταν 300 (1500 πλην 1200).

Στόχος σας είναι να γράψετε ένα πρόγραμμα που θα δέχεται τους χρόνους N αγροτών (1 <= N <= 5000) που αρμέγουν N αγελάδες και θα τυπώνεται (σε δευτερόλεπτα):

Το μέγιστο χρόνο συνεχούς αρμέγματος.

Το μέγιστο χρόνο που καμιά αγελάδα δεν αρμεγόταν.

Δεδομένα Εισόδου

Στη γραμμή 1 υπάρχει ένας ακέραιος αριθμός που αντιστοιχεί στο N.

Στη συνέχεια υπάρχουν N ζευγάρια ακεραίων αριθμών που αντιστοιχούν στην έναρξη και τη λήξη του αρμέγματος για κάθε αγρότη. Οι αριθμοί είναι μικρότεροι του 1000000

Δεδομένα Εξόδου

Μια γραμμή που περιέχει 2 ακεραίους αριθμούς.

Παράδειγμα Εισόδου

3

300 1000

700 1200

1500 2100

Παράδειγμα Εξόδου

900 300

Επεξήγηση

Το πρόβλημα λύνεται με ταξινόμηση. Αποθηκεύουμε τα ζευγάρια των αριθμών σε 2 μονοδιάστατους πίνακες και τους ταξινομούμε. ΟΧΙ παράλληλη ταξινόμηση. Στη συνέχεια ξεκινώ από τη 2η θέση του πίνακα και ελέγχω αν ο χρόνος έναρξης στη θέση i είναι μικρότερος από τον χρόνο λήξης στη θέση i-1. Αν ισχύει αυτό τότε σημαίνει ότι υπάρχει οι 2 αγρότες δούλευαν ταυτόχρονα. Αν όχι υπολογίζω τους χρόνους και ελέγχω αν είναι μέγιστοι.

Ενδεικτική Λύση

```
#include<iostream>
#include<algorithm>
#define size 5000
using namespace std;
int main(){
  int N,i,checkAll=0;
  long int startMax,max1,temp,max2=0;
  long int start[size],finish[size];
   cin>>N;
   for(i=0;i<N;i++){
    cin>>start[i]>>finish[i];
   }
   sort(start, start+N);
   sort(finish, finish+N);
   startMax=start[0];
   max1=finish[0]-start[0];
   for(i=1;i<N;i++){
      if(start[i]>finish[i-1]){
        temp=finish[i-1]-startMax;
        if(temp>max1)
          max1=temp;
        temp=start[i]-finish[i-1];
        if(temp>max2)
          max2=temp;
        startMax=start[i];
```

```
        checkAll=1;
      }
    }
  if(checkAll==0)
    max1=finish[N-1]-start[0];
 cout<<max1<<" "<<max2<<endl;
 return 0;
}
```

Test Cases

Κάθε αγρότης αρμέγει μία αγελάδα, υπάρχουν όμως αγρότες που μπορεί να αρμέγουν τις αγελάδας τους ταυτόχρονα. Μπορεί επίσης να υπάρχει άρμεγμα των αγελάδων για σε όλο το διάστημα. Ο έλεγχος αυτός γίνεται στο τεστ 2.

Υπάρχει περίπτωση να έχουμε μόνο έναν αγρότη. Αυτός ο έλεγχος γίνεται στο τεστ 3. Τέλος στο τεστ 5 ελέγχουμε ότι το πρόγραμμα μπορεί να περάσει τον έλεγχο του μέγιστου αριθμού αγροτών.

A/A	Input	Output
1	3 300 1000 700 1200 1500 2100	900 300
2	2 100 200 200 300	200 0
3	1 100 200	100 0
4	6 100 200 200 400 400 800 800 1600 50 100	1550 100

	1700 3200		
5	Ένα αρχείο με 5000 αγρότες. Είναι διαθέσιμο σε ηλεκτρονική μορφή		

5.5.5 Στοίβες (Stacks)

Οι στοίβες είναι δομές δεδομένων που ακολουθούν την αρχή LIFO – Last In First Out. Σκεφτείτε να έχουμε μια στοίβα από πιάτα. Το πιάτο στην κορυφή είναι το μοναδικό στο οποίο έχουμε πρόσβαση. Για να πάρουμε ένα πιάτο που βρίσκεται πιο κάτω θα πρέπει να αφαιρέσουμε (pop) όλα τα πιάτα που βρίσκονται από πάνω του. Υπάρχουν στοίβες που είναι στατικές και μπορούν να υλοποιηθούν με τη χρήση ενός πίνακα και δυναμικές που υλοποιούνται με τη χρήση pointers. Το πλεονέκτημα των στατικών είναι ότι είναι εύκολη η υλοποίηση τους και το μειονέκτημα τους ότι πρέπει να είναι γνωστό το πλήθος των δεδομένων που θα εισαχθούν στη στοίβα. Το πλεονέκτημα των δυναμικών είναι ότι μπορούμε να τις χρησιμοποιήσουμε ανεξαρτήτως μεγέθους και το μειονέκτημα τους ότι είναι δύσκολη η υλοποίηση τους [04].

Άσκηση 1

Εκφώνηση

Ένας ρομποτικός βραχίονας είναι υπεύθυνος για την τοποθέτηση κιβωτίων μέσα στην αποθήκη ενός πλοίου. Τα κιβώτια έχουν όλα το ίδιο μέγεθος και για να χωρέσουν στην αποθήκη πρέπει να μπει το ένα πάνω από το άλλο. Κάθε κιβώτιο έχει 2 κωδικούς. Ο πρώτος αντιστοιχεί σε ένα κεφαλαίο γράμμα του αγγλικού αλφαβήτου και ο δεύτερος σε έναν μονοψήφιο ακέραιο αριθμό. Η εισαγωγή των κιβωτίων γίνεται ως εξής:

Ο βραχίονας δέχεται μια εντολή στη μορφή X Y Z όπου X ο πρώτος κωδικός του δέματος και Y ο δεύτερος κωδικός του δέματος. Το Z μπορεί να είναι ο αριθμός 1 ή ο αριθμό 2 και αναφέρεται στον κωδικό ο οποίος θα ληφθεί υπόψη για την εισαγωγή του κιβωτίου.

0Το κάθε κιβώτιο πρέπει να τοποθετείται πάνω από κιβώτιο με τον ίδιο Z κωδικό. Αν δεν υπάρχει άλλο κιβώτιο με τον ίδιο κωδικό τότε το κιβώτιο πηγαίνει στον πάτο της αποθήκης.

Είσοδος

Γραμμή 1: Ο αριθμός N (0<N<=10000) που αντιστοιχεί στις εντολές του βραχίονα

Γραμμές 2..N+1: Κάθε γραμμή περιέχει μια εντολή στη μορφή X, Y, Z

Έξοδος

Τα κιβώτια με την σειρά που θα εκφορτωθούν από το πλοίο. Για το κάθε κιβώτιο θα γράφονται οι κωδικοί X και Y.

Παράδειγμα Εισόδου

5

A 3 1

A 2 2

A 8 1

K 2 2

Z 3 1

Παράδειγμα Εξόδου

A 8

A 3

K 2

A 2

Z 3

Επεξήγηση

Το πρόβλημα εξετάζει τη χρήση των στοιβών και των διαδικασιών pop και push. Το πρόγραμμα χρειάζεται 2 στοίβες. Η πρώτη για να αποθηκεύει τα υπόγεια και η δεύτερη για να αποθηκεύει τα κιβώτια που θα βγαίνουν από την πρώτη και θα πρέπει μετά να ξαναμπούν.

Ενδεικτική Λύση

```
#include<iostream>
using namespace std;
struct node{
char X;
int Y;
node* next;
};
struct node* top1=NULL;
struct node* top2=NULL;
void push1(char A,int B){
   node* temp=new node;

   temp->X=A;
   temp->Y=B;
   temp->next=top1;
   top1=temp;
};
void push2(char A,int B){
   node* temp=new node;
   temp->X=A;
   temp->Y=B;
   temp->next=top2;
   top2=temp;
};

void pop1(){
   if(top1==NULL)
```

```
    return;
   top1=top1->next;
};

void pop2(){
   if(top2==NULL)
    return;
   top2=top2->next;
};

int main(){
  char X;
  int N,Y,Z,i;
  node* temp1;
  node* temp2;

  cin>>N;
  cin>>X>>Y>>Z;
  push1(X,Y);
  for(i=1;i<N;i++){
   cin>>X>>Y>>Z;
   if(Z==1){
    temp1=top1;
    temp2=top2;
    while(temp1!=NULL){
     if(temp1->X==X){
      push1(X,Y);
      while(top2!=NULL){
       push1(top2->X,top2->Y);
       pop2();
      }
      goto down;
     }// if
     else{
```

```
      push2(temp1->X,temp1->Y);
      temp1=temp1->next;
      pop1();
     }
    }// while
    push1(X,Y);
    while(top2!=NULL){
     push1(top2->X,top2->Y);
     pop2();
    }
   }// if
   else{
    temp1=top1;
    temp2=top2;
    while(temp1!=NULL){
     if(temp1->Y==Y){
      push1(X,Y);
      while(top2!=NULL){
       push1(top2->X,top2->Y);
       pop2();
      }
      goto down;
     }// if
     else{
      push2(temp1->X,temp1->Y);
      temp1=temp1->next;
      pop1();
     }
    }// while
    push1(X,Y);
    while(top2!=NULL){
     push1(top2->X,top2->Y);
     pop2();
    }
```

```
    }// else
    down:;
   }// for
   while(top1!=NULL){
    cout<<top1->X<<" "<<top1->Y<<endl;
    top1=top1->next;
   }
   return 0;
}
```

Test Cases

Οι έλεγχοι πρέπει να γίνονται και για τους 2 κωδικούς. Επίσης στο τελευταίο τεστ ελέγχεται το αριθμητικό όριο που θέσαμε για τις εντολές.

A/A	Input	Output
1	5 A 3 1 A 2 2 A 8 1 K 2 2 Z 3 1	A 8 A 3 K 2 A 2 Z 3
2	3 Z 1 1 Z 2 1 Z 4 1	Z 4 Z 2 Z 1
3	10 A 1 1 D 2 2 K 1 2 S 7 1 L 6 1 L 4 1	Z 1 K 1 A 1 D 1 D 2 D 2 S 7

	D 2 2 D 1 1 Z 1 2 Z 3 2	L 4 L 6 Z 3	
4	Ένα αρχείο με 100 εντολές. Είναι διαθέσιμο σε ηλεκτρονική μορφή		
5	Ένα αρχείο με 10000 εντολές. Είναι διαθέσιμο σε ηλεκτρονική μορφή		

Άσκηση 2

Εκφώνηση

Η Αριάδνη έχει πάρει το πλοίο για να πας να συναντήσει το φίλο της τον Κυριάκο. Για να περάσει την ώρα της στην διαδρομή παρατηρεί ποια αυτοκίνητα μπαίνουν και πόσα αυτοκίνητα φεύγουν σε κάθε λιμάνι που κάνει στάση το πλοίο.

Αν στο συγκεκριμένο πλοίο το πρώτο αυτοκίνητο που βγαίνει κάθε φορά είναι αυτό που μπήκε τελευταίο, το δεύτερο αυτοκίνητο που βγαίνει είναι αυτό που μπήκε προτελευταίο κ.ο.κ., βοήθησε την Αριάδνη να βρει ποια αυτοκίνητα υπάρχουν στο πλοίο όταν αυτό φτάνει στο λιμάνι που κατεβαίνει.

Δεδομένα Εισόδου

Η πρώτη γραμμή περιέχει έναν αριθμό N <= 10002, τον αριθμό των λιμανιών που συναντά η Αριάδνη πριν τον τελικό της προορισμό. Κάθε μία από τις N επόμενες γραμμές αντιστοιχεί σε καθε λιμάνι. Για κάθε λιμάνι υπάρχουν 2 ακέραιοι, ο αριθμός των αυτοκινήτων που βγήκαν από το πλοίο Xi και ο αριθμός των αυτοκινήτων που μπήκαν στο πλοίο Yi. Ακολουθούν Yi λέξεις αποτελούμενες από 42 ή λιγότερους λατινικούς χαρακτήρες που αντιστοιχούν στο μοντέλο του κάθε αυτοκινήτου που μπαίνει στο πλοίο με τη σειρά που αυτά μπήκαν.

Δεδομένα Εξόδου

Πρέπει να υπάρχει μια λίστα με τα αυτοκίνητα που υπάρχουν στο πλοίο, ταξινομημένη αλφαβητικά, ένα αυτοκίνητο σε κάθε γραμμή.

Παράδειγμα εισόδου

3

0 4 Audi Ford Fiat Toyota

2 3 BMW Suzuki Honda

4 2 Mercedes Citroen

Παράδειγμα εξόδου

Audi

Citroen

Mercedes

Επεξήγηση

Το πρόβλημα είναι σχετικά απλό και χρειάζεται μόνο μια στοίβα. Η μόνη δυσκολία είναι τα περιεχόμενα της στοίβας πρέπει να ταξινομηθούν. Για το λόγο χρησιμοποιώ ένα vector. Στην ενδεικτική λύση αυτού του προβλήματος έχω χρησιμοποιήσει βιβλιοθήκες από το STL (stack, algorithm) και είναι φανερό ότι το μέγεθος του προγράμματος έχει μειωθεί κατά πολύ. Να σημειωθεί ότι η χρήση του STL επιτρέπετε και στους διεθνείς διαγωνισμούς.

Ενδεικτική Λύση

```
#include <iostream>
#include <string>
#include <stack>
#include <vector>
#include <algorithm>
using namespace std;

int main(){
 stack<string> s;
 vector<string> v;
 int N,ipop,ipush,i,j,k;
  string car;
 cin>>N;
 for(i=1;i<=N;i++){
      cin>>ipop;
      cin>>ipush;
      for(j=1;j<=ipop;j++)
       s.pop();
      for(j=1;j<=ipush;j++){
         cin>>car;
         s.push(car);         }

              }
    k=s.size();
    for(i=1;i<=k;i++){
    v.push_back(s.top());
    s.pop();}

sort (v.begin(), v.begin()+k);
for(i=0;i<k;i++)
 cout<<v[i]<<"\n";
 return 0;}
```

Test Cases

Στο τεστ 2 δεν καταβαίνει κανένα αυτοκίνητο, ενώ στο τεστ 3 δεν υπάρχει αυτοκίνητο μέσα στο πλοίο. Επίσης στο τελευταίο τεστ ελέγχεται το αριθμητικό όριο που θέσαμε για τον αριθμό των λιμανιών.

A/A	Input	Output
1	3 0 4 Audi Ford Fiat Toyota 2 3 BMW Suzuki Honda 4 2 Mercedes Citroen	Audi Citroen Mercedes
2	3 0 4 Audi Ford Fiat Toyota 0 3 BMW Suzuki Honda 0 2 Mercedes Citroen	Audi BMW Citroen Fiat Ford Honda Mercedes Suzuki Toyota
3	2 0 4 Audi Ford Fiat Toyota 4 0	
4	Ένα αρχείο με 100 λιμάνια. Είναι διαθέσιμο σε ηλεκτρονική μορφή	
5	Ένα αρχείο με 10002 λιμάνια Είναι διαθέσιμο σε ηλεκτρονική μορφή	

5.5.6 Γράφοι (Graphs)

Οι γράφοι προσφέρουν μια χρήσιμη μέθοδο για τη διατύπωση πολλών προβλημάτων σε:

- Δίκτυα και συστήματα τηλεπικοινωνιών
- Χάρτες – επιλογή δρομολογίων

- Προγραμματισμό εργασιών (scheduling)

Ένας γράφος αποτελείται από ένα σύνολο κορυφών (vertices) και ένα σύνολο ακμών (edges). Η αναπαράσταση ενός γράφου μπορεί να γίνει με τη χρήση ενός πίνακα ή μιας λίστας γειτνίασης [04]. Οι 3 ασκήσεις που ακολουθούν αναφέρονται στα πιο κάτω θέματα:

1. Δημιουργία πίνακα γειτνίασης
2. Διάσχιση γράφου
3. Εύρεση συντομότερου μονοπατιού

Άσκηση 1

Εκφώνηση

Η Αριάδνη έχει πιάσει δουλειά στο γραφείο ενός πολιτικού μηχανικού. Ο πολιτικός μηχανικός έχει αναθέσει στην Αριάδνη να του εμφανίσει σε μορφή πίνακα τις συνδέσεις μεταξύ των κολώνων του κτιρίου.

Δεδομένα Εισόδου

Γραμμή 1: Ένας ακέραιος αριθμός Α (Α<=100)που αντιστοιχεί στον αριθμό των κολώνων του κτιρίου και ένας αριθμός Β (Β<=300) που αντιστοιχεί στον αριθμό των συνδέσεων.

Γραμμές 2 ..Β+1: Σε κάθε γραμμή υπάρχει ένα ζευγάρι αριθμών που αντιστοιχεί στο ζεύγος των κολώνων που είναι συνδεδεμένες.

Δεδομένα Εξόδου

Ένας πίνακας που σημειώνει με 1 την ύπαρξη σύνδεσης και με 0 την απουσία της

Παράδειγμα Εισόδου

5 6

1 3

2 3

3 4

4 5

5 2

Παράδειγμα Εξόδου

0 0 1 0 0

0 0 1 0 0

0 0 0 1 0

0 0 0 0 1

0 1 0 0 0

Επεξήγηση

Το πρόβλημα είναι πολύ απλό και έχει σκοπό την εισαγωγή των μαθητών στην έννοια του πίνακα γειτνίασης

Ενδεικτική Λύση

```
#include <iostream>
#define N 100
using namespace std;
```

```
int main(){
int a,b,i,j,x,y;
int adj[N+1][N+1]={};

cin>>a>>b;
 for(i=0;i<b;i++){
   cin>>x>>y;
   adj[x][y]=1;
            }

 for(i=1;i<=a;i++){
  for(j=1;j<=a;j++)
   cout<<adj[i][j]<<" ";
  cout<<endl;
 }

  return 0;
}
```

Test Cases

Στο τεστ 2 δεν υπάρχει καμία σύνδεση, ενώ στο τεστ 3 υπάρχει σύνδεση μεταξύ όλων των κολώνων. Στο τελευταίο τεστ ελέγχεται το αριθμητικό όριο που θέσαμε για τον αριθμό των κολώνων και των συνδέσεων.

A/A	Input	Output
1	5 6 1 3 2 3 3 4 4 5 5 2	0 0 1 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 1 0 0 0

2	3 0	0 0 0 0 0 0 0 0 0	
3	3 6 1 2 2 1 1 3 2 3 3 1 3 2	0 1 1 1 0 1 1 1 0	
4	Ένα αρχείο με 100 κολώνες και 100 συνδέσεις. Είναι διαθέσιμο σε ηλεκτρονική μορφή		
5	Ένα αρχείο με 100 κολώνες και 300 συνδέσεις. Είναι διαθέσιμο σε ηλεκτρονική μορφή		

Άσκηση 2

Εκφώνηση

Στη Αριάδνη αρέσει να ταξιδεύει με το τρένο. Κάθε φορά που επισκέπτεται μία χώρα προσπαθεί να επισκεφθεί τις διάφορες πόλεις με το τρένο. Δυστυχώς όμως δεν είναι πάντοτε εφικτό να επισκέπτεται την πόλη που επιθυμεί. Βοηθήστε την Αριάδνη φτιάχνοντας ένα πρόγραμμα που θα δέχεται το δρομολόγιο των τρένων, την πόλη αναχώρησης και την πόλη άφιξης και θα ενημερώνει την Αριάδνη αν είναι εφικτή ή όχι η επίσκεψη της. Σε περίπτωση θετικής απάντησης να τυπώνονται και οι ενδιάμεσες πόλεις από όπου περνά το τρένο.

Δεδομένα Εισόδου

Οι πόλεις είναι αριθμημένες με συνεχόμενους ακέραιους αριθμούς από το μηδέν μέχρι το N (N<=50). Στην αρχή δίνονται 3 ακέραιοι αριθμοί που αντιστοιχούν στην πόλη αναχώρησης, στην πόλη άφιξης και στον αριθμό των δρομολογίων. Στη συνέχεια ακολουθεί το δρομολόγιο των τρένων που αποτελείται από ζευγάρια ακεραίων αριθμών (A B) που υποδηλώνει την ύπαρξη δρομολογίου από την πόλη A στην πόλη B

Σημείωση: Η ύπαρξη δρομολογίου από την πόλη Α στην Β δεν υποδηλώνει και την ύπαρξη δρομολογίου από την Β στην Α.

Δεδομένα Εξόδου

Το πρόγραμμα θα τυπώνει τον αριθμό -1 σε περίπτωση που δεν υπάρχει σύνδεση μεταξύ των δύο πόλεων, ενώ στην αντίθετη περίπτωση θα τυπώνει όλες τις πόλεις ξεκινώντας από την πόλη αναχώρησης μέχρι και την πόλη άφιξης.

Παράδειγμα εισόδου 1
1 4 6
0 1
1 5
4 1
2 3
1 2
0 4
Παράδειγμα εξόδου 1
-1
Παράδειγμα εισόδου 2
2 1 6
5 1
6 2
2 4
4 5
0 1
1 3
Παράδειγμα εξόδου 2
2
4
5
1

Επεξήγηση

Σε αυτή την άσκηση έχουμε ένα πρόβλημα διάσχισης. Το πρόγραμμα δημιουργεί τον πίνακα γειτνίασης και στη συνέχεια τρέχει έναν αλγόριθμο διάσχισης (DFS) με αρχή την πόλη αφετηρία. Αν το πρόγραμμα φτάσει μέχρι την πόλη σταθμός τότε τυπώνεται η διαδρομή, διαφορετικά τυπώνεται -1.

Ενδεικτική Λύση

```
#include <vector>
#include <iostream>
#define N 50
using namespace std;
void dfs(int start);
 vector< vector<int> > adj;
  vector<int> v,cities;
   int a,b,counter=0,found=0,dfs1=0;
int main(){

 int i,j,d;

  for(i=0;i<N;i++){
     adj.push_back(vector<int> ());
     v.push_back(0);
     cities.push_back(0);
     }
    cin>>a>>b>>d;
   for (int k=1;k<=d;k++){
    cin>>i>>j;
    adj[i].push_back(j);
              }

 dfs(a);
 if(found==0)
  cout<<-1<<endl;
```

```
 else{
    dfs1=1;
  cout<<a<<endl;
  j=counter;
  for(a=1;a<j;a++){
   for(i=0;i<v.size();i++)
    v[i]=0;
   found=0;
   counter=0;
   dfs(cities[a]);
   if(found!=0)
   cout<<cities[a]<<endl;
    }
}
   return 0;
}

void dfs(int start){
   int w,size;
   v[start]=1;
if(dfs1==0)
   cities[counter]=start;
   counter++;
   if(start==b)
    found=1;
   else{
   size=adj[start].size();
   for (w=0;w<size;w++){
      if ((v[adj[start][w]]==0)){
          dfs(adj[start][w]);}
    }
    }
}
```

Test Cases

Η μεγάλη δυσκολία στη δημιουργία των test cases για αυτή την άσκηση ήταν ότι έπρεπε να δημιουργήσω μοναδικές διαδρομές από την πόλη αναχώρησης μέχρι τον προορισμό.

A/A	Input	Output
1	1 4 6 0 1 1 5 4 1 2 3 1 2 0 4	-1
2	2 1 6 5 1 6 2 2 4 4 5 0 1 1 3	2 4 5 1
3	1 20 19 18 19 1 2 2 3 3 4 4 5 5 6 6 7 7 8 8 9 9 10 10 11 11 12 12 13	1 2 3 4 5 6 7 8 9 10 11 12 13 14

	13 14 14 15 15 16 16 17 17 18 19 20	15 16 17 18 19 20	
4	Ένα αρχείο με 50 πόλεις και 100 δρομολόγια. Είναι διαθέσιμο σε ηλεκτρονική μορφή		
5	Ένα αρχείο με 50 πόλεις και 1000 δρομολόγια. Είναι διαθέσιμο σε ηλεκτρονική μορφή		

Άσκηση 3

Εκφώνηση

Σε ένα ρυάκι υπάρχουν n εμπορικοί σταθμοί που αριθμούνται από 1 έως n. Σε κάθε σταθμό i μπορείς να ενοικιάσεις κανό το οποίο είναι δυνατό να επιστραφεί σε οποιοδήποτε άλλο σταθμό j > i. Ζητείται να γραφεί πρόγραμμα το οποίο δέχεται ως είσοδο ένα πίνακα C(i, j) που αντιπροσωπεύει το κόστος ενοικιάσεως από τον σταθμό i έως το σταθμό j (για κάθε 1<=i, j <=n) και επιστρέφει τη πιο φθηνή διαδρομή ενοικιάσεων από το σταθμό 1 έως το σταθμό n. Μπορείτε να υποθέσετε ότι C(i,i)=0, και ότι δεν επιτρέπεται να πάτε αντίθετα στο ρεύμα.

Δεμένα εισόδου

Η πρώτη γραμμή εισόδου αποτελείται από ένα θετικό ακέραιο αριθμό n (n<=30) που αντιπροσωπεύει τον αριθμό των σταθμών. Οι επόμενες n γραμμές αποτελούνται από n ακέραιους αριθμούς η κάθε μια, οι οποίοι διαχωρίζονται με κενό διάστημα και έχουν τιμές μικρότερες από 1000.

Δεδομένα Εξόδου

Το πρόγραμμα πρέπει να επιστρέφει δύο γραμμές εξόδου. Η πρώτη γραμμή εξόδου θα πρέπει να αποτελείται από ένα θετικό ακέραιο αριθμό m (m<=100) που αντιπροσωπεύει τον αριθμό των ενοικιάσεων της πιο φθηνής διαδρομής. Η δεύτερη γραμμή θα πρέπει να αποτελείται από m+1 ακέραιους αριθμούς που υποδηλώνουν τους σταθμούς της πιο φθηνής διαδρομής από τον σταθμό 1 έως το σταθμό n.

Παράδειγμα Εισόδου

4

0 2 3 7

0 0 2 4

0 0 0 2

0 0 0 0

Παράδειγμα Εξόδου

3

1 3 4

Επεξήγηση

Σε αυτή την άσκηση έχουμε έναν κατευθυνόμενο γράφο και πρέπει να επιλέγουμε το συντομότερο μονοπάτι (shortest path). Για να το πετύχω αυτό χρησιμοποιώ τον αλγόριθμο του Dijkstra.

Ενδεικτική Λύση

```
#include<iostream>
#include<vector>
#include<stack>
# define max 32000
using namespace std;
int main(){
   int i,j,n,source,w,temp,u,counter=1;
   vector< vector<int> > adj;
   vector<int> visited;
   vector<int> pred;
   vector<int> dist;
   stack<int> path;

   cin>>n;
   for(i=0;i<n;i++){
     adj.push_back(vector<int> ());
```

```
  visited.push_back(0);
  pred.push_back(0);
  dist.push_back(0);}

for(j=0;j<n;j++)
  adj[j].push_back(0);

for(i=0;i<n;i++)
 for(j=0;j<n;j++){
   cin>>adj[i][j];
   if((adj[i][j]==0)&&(i!=j))
     adj[i][j]=max;
     }
source=0;
for(i=0;i<n;i++)
 dist[i]=adj[source][i];
visited[source]=1;
dist[source]=0;
for(i=0;i<(n-2);i++){
   temp=max;
   for(j=0;j<n;j++)
     if((dist[j]<temp)&&(visited[j]==0)){
        temp=dist[j];
        u=j;
     }
visited[u]=1;
for(w=0;w<n;w++)
   if(visited[w]==0)
     if(adj[u][w]!=max)
        if(dist[w]>(dist[u]+adj[u][w])){
           dist[w]=dist[u]+adj[u][w];
           pred[w]=u;
        }
}
```

```
  i=n-1;
  while(i!=0){
     path.push(i+1);
     i=pred[i];
     counter++;}
     path.push(1);
  cout<<counter<<"\n";
  i=path.size();
  for(j=1;j<=i;j++){
   cout<<path.top()<<" ";
   path.pop();}
   return 0;
}
```

Test Cases

Σε αυτή την άσκηση πρέπει να είμαστε προσεκτικοί όταν δημιουργούμε τα test cases, να έχουμε οπωσδήποτε μια διαδρομή από την αφετηρία στον τερματισμό. Επίσης πρέπει να έχουμε περισσότερες από μια διαδρομές, ώστε να ελέγχουμε ότι ο μαθητής εφάρμοσε τον αλγόριθμο για τη συντομότερη διάδρομή.

A/A	Input	Output
1	4 0 2 3 7 0 0 2 4 0 0 0 2 0 0 0 0	3 1 3 4
2	5 0 1 3 5 7 0 0 1 4 7 0 0 0 1 7 0 0 0 0 1 0 0 0 0 0	5 1 2 3 4 5

3	5 0 9 3 5 7 0 0 1 4 7 0 0 0 1 7 0 0 0 0 1 0 0 0 0 0	4 1 3 4 5	
4	10 0 9 3 5 7 1 10 8 9 0 0 1 4 7 33 2 40 40 0 0 0 1 7 31 4 10 22 0 0 0 0 1 11 312 22 123 0 0 0 0 0 5 54 55 3 645 0 0 0 0 0 0 3 43 44 33 0 0 0 0 0 0 0 111 32 11 0 0 0 0 0 0 0 0 12 1 0 0 0 0 0 0 0 0 0 100 0 0 0 0 0 0 0 0 0 0	3 1 6 10	
5	Ένα αρχείο με 30 σταθμοί και 100 ενοικιάσεις. Είναι διαθέσιμο σε ηλεκτρονική μορφή		

Κεφάλαιο 6
Έλεγχος Συστήματος

Η φάση του ελέγχου είναι πολύ βασική για την ομαλή λειτουργία του συστήματος. Είναι γενικά παραδεκτό ότι αυτός που δημιουργεί το σύστημα δε θα πρέπει να κάνει και τον έλεγχο. Για το λόγο αυτό έδωσα πρόσβαση στο σύστημα σε 2 μαθητές και τους παρακάλεσα να δοκιμάσουν και να καταγράψουν τις δοκιμές που έκαναν σε έναν πίνακα. Οι έλεγχοι που πραγματοποίησα εγώ και οι 2 μαθητές καταγράφονται και παρουσιάζονται στον πιο κάτω πίνακα.

6.1 Πίνακας ελέγχων

Το σύστημα έχει περάσει όλους τους ελέγχους που αναγράφονται στον πιο κάτω πίνακα.

Α/Α	Έλεγχος που πραγματοποιήθηκε
1	Λανθασμένο Login
2	Εγγραφή με ελλειπή στοιχεία

3	Εγγραφή με username μεγαλύτερο των 20 χαρακτήρων
4	Εγγραφή με λανθασμένο password
5	Υποβολή πηγαίου κώδικα που δεν ανταποκρίνεται στην επιλογή γλώσσας
6	Υποβολή κώδικα με συντακτικά λάθη
7	Υποβολή λύσης που δεν πετυχαίνει κανένα test case
8	Υποβολή λύσης που πετυχαίνει μερικά test cases
9	Υποβολή λύσης που πετυχαίνει όλα τα test cases
10	Υποβολή λύσης που ξεπερνά το χρονικό όριο
11	Υποβολή λύσης χωρίς να επιλεγεί πηγαίος κώδικας
12	Επιλογή κατηγορίας από τη θεωρία και εμφάνιση των σωστών αρχείων
13	Ανάρτηση προβλήματος χωρίς εκφώνηση
14	Ανάρτηση προβλήματος χωρίς τίτλο
15	Ανάρτηση προβλήματος χωρίς test cases
16	Ανάρτηση προβλήματος με εκφώνηση που δεν είναι text files
17	Εμφάνιση του σωστού προβλήματος για διόρθωση
18	Διόρθωση προβλήματος χωρίς τίτλο
19	Διαγραφή προβλήματος
20	Φόρτωση προβλημάτων για συγκεκριμένο χρήστη
21	Φόρτωση όλων των προβλημάτων για το χρήστη administrator
22	Ανάρτηση θεωρίας χωρίς να επιλεγεί αρχείο
23	Ανάρτηση θεωρίας χωρίς τίτλο
24	Διαγραφή επιλεγμένου αρχείου θεωρίας

25	Αλλαγή τύπου χρήστη
26	Αλλαγή username, password και email χρήστη
27	Διαγραφή χρήστη

Κεφάλαιο 7
Επίλογος

Όπως συμβαίνει με όλα τα συστήματα έτσι και στο σύστημα «Αριάδνη» αντιμετώπισα διάφορα προβλήματα κατά τη δημιουργία του. Επίσης υπάρχουν διάφορες αναβαθμίσεις που προτείνονται τόσο από εμένα όσο και από άτομα που δοκίμασαν το σύστημα.

7.1 Προβλήματα

Το κυριότερο πρόβλημα που αντιμετώπισα ήταν ότι η δημιουργία του συστήματος ήταν «one man show», δηλαδή έπρεπε να επωμιστώ όλα τα στάδια από την αρχή μέχρι το τέλος. Αυτό δεν κάτι συνηθισμένο όταν δημιουργείται ένα σύστημα.

Άλλη δυσκολία που αντιμετώπισα ήταν οι ελάχιστες γνώσεις που είχα στη PHP και MySQL. Αυτός όμως ήταν και ένας από τους λόγους που επέλεξα αυτό το θέμα, ώστε να αναγκάσω τον εαυτό μου να μάθει αυτό το αντικείμενο. Ήταν έναν πολύ ενδιαφέρον «ταξίδι» αφού έμαθα πολλά, μέσα από βιβλία και τον παγκόσμιο ιστό. Πολλές φορές έκανα λάθη και αναγκάστηκα να επιστρέψω πίσω για να τα διορθώσω. Είμαι σίγουρος ότι πολλά πράγματα θα μπορούσαν να

γίνουν διαφορετικά και με πιο σύντομο τρόπο, αλλά είμαι αρκετά ευχαριστημένος με τον τρόπο που δουλεύει το σύστημα και στον τρόπο που με τον οποίο ανταποκρίθηκε στους διάφορους ελέγχους.

Ένα άλλο πρόβλημα ήταν η αδυναμία να ελέγξω το σύστημα με πολλούς χρήστες ενωμένους σε αυτό. Ο κυριότερος λόγος ήταν η έλλειψη χρόνου αλλά και μικρές δυνατότητες του υπολογιστή που φιλοξενεί το σύστημα. Υπάρχουν σχέδια να το μεταφέρω σε καλύτερο υπολογιστή τουλάχιστον προσωρινά μέχρι να βρω μια μόνιμη φιλοξενία.

Τέλος να αναφέρω την περιορισμένη βιβλιογραφία που υπάρχει στην κατασκευή online judge συστημάτων. Οι περισσότερες αναφορές που βρήκα ήταν από παλιές διατριβές σε πανεπιστήμια του εξωτερικού.

7.2 Μελλοντικές Αναβαθμίσεις

Οι παρακάτω προτάσεις για αναβαθμίσεις προέρχονται από παρατηρήσεις δικές μου αλλά και ατόμων που χρησιμοποίησαν το σύστημα. Κάποιες από αυτές είναι σημαντικές (critical) και θα πρέπει να υλοποιηθούν πριν γίνει το σύστημα public. Άλλες είναι λιγότερο σημαντικές αλλά η υλοποίηση τους θα βοηθούσε την καλύτερη λειτουργία του συστήματος.

Critical Αναβαθμίσεις

1. Παραμετροποίηση των ορίων του χρόνου εκτέλεσης και της μνήμης για κάθε πρόβλημα. Στα προβλήματα του επιπέδου της ολυμπιάδας πληροφορικής αυτοί οι δύο παράγοντες είναι σημαντικοί.

2. Θα πρέπει να δοθεί μεγαλύτερη σημασία στα θέματα ασφαλείας. Το σύστημα δεν έχει δοκιμαστεί σε διάφορες κακόβουλες χρήσεις (π.χ. sql injection, DoS). Σε αυτό το σημείο θα ήταν χρήσιμο να συμβουλευτώ κάποια άτομα με εμπειρία στον τομέα της ασφάλειας.

3. Δυνατότητα εισαγωγής εικόνων στις εκφωνήσεις των ασκήσεων

Χρήσιμες Αναβαθμίσεις

1. Κάποια άτομα εισηγήθηκαν να αλλάξει το interface του συστήματος και να χρησιμοποιηθεί για αυτόν τον σκοπό το Joomla.

2. Δυνατότητα διόρθωσης/ διαγραφής των test cases σε μια άσκηση.

3. Διαφορετική βαρύτητα στη βαθμολόγηση για το κάθε test case.

4. Αξιολόγηση ασκήσεων σε Java

5. Δυνατότητα δημιουργίας διαγωνισμών

Σαν καταληκτικό σχόλιο θα ήθελα να εκφράσω την πίστη και την αισιοδοξία μου ότι το σύστημα αυτό με την υλοποίηση των κρίσιμων αναβαθμίσεων και αφού περάσει περαιτέρω ελέγχους θα μπορεί να δοθεί στη διάθεση των μαθητών/φοιτητών της Κύπρου και να είναι ένα χρήσιμο εργαλείο στην μύηση στον μαγικό κόσμο του προγραμματισμού.

Βιβλιογραφία

[01] Βασίλειος Βεσκούκης, (2000), Αρχές Τεχνολογίας Λογισμικού Τόμος Α΄, Ελληνικό Ανοικτό Πανεπιστήμιο

[02] Κάκας Αντώνης, Ανδρέου Ανδρέας, Ρετάλης Συμεών (2004) Ανάλυση και Σχεδίαση Πληροφοριακών Συστημάτων, Πανεπιστήμιο Κύπρου

[03] Luke Weling, Laura Thomson (2001), Ανάπτυξη Web Εφαρμογών με PHP και MySQL

[04] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Cliford Stein (2005), Introduction to Algorithms

[05] PHP 5 Tutorials, http://www.w3schools.com/php/default.asp

[06] SQL Tutorials, http://www.w3schools.com/sql/default.asp

[07] Batch files and command line parameters, http://arstechnica.com/civis/viewtopic.php?f=17&t=1051090

Παράρτημα Α
Ερωτηματολόγιο

Α.1 Ερωτηματολόγιο Συλλογής Πληροφοριών

Το παρακάτω ερωτηματολόγιο ετοιμάστηκε για να συλλέξω πληροφορίες στο στάδιο της προκαταρκτικής έρευνας. Δόθηκε σε τρεις (3) team leaders κατά τη διάρκεια της Βαλκανιάδα Πληροφορικής το 2012 στο Βελιγράδι. Τα άτομα αυτά ήταν:

- Dr. Constantin Gălăţan (Romania)
- Δρ. Γιάννης Βογιατζής (Ελλάδα)
- Dr. Goran Sukovic (Montenegro)

Questionnaire Cover Letter

Dear sir/madam,

I am a postgraduate student at Open University of Cyprus (OUC). I am a High School teacher of Information Technology at Ministry of Education of Cyprus. I am also coaching students for the Internationals Olympiads of Informatics.

For my master thesis I am researching methods for boosting my students' knowledge at Programming by using an auto grater system. I am interested both for technical and educational issues.

Your involvement at your country's similar projects concerning informatics competitions and your experience at such matters will be very valuable for my survey.

I am attaching a questionnaire that will require approximately 10-15 minutes to complete. If you need further explanations do not hesitate to contact me, or my supervisor Dr. Maria Andreou (andreou@ouc.ac.cy).

Thank you in advance for your help.

Yours Sincerely

Pavlos Pavlikas

Questionnaire

Section 1: Respondent Information

1.1 Name:

1.2 Country:

1.3 Job Title:

Section 2: System Usefulness

2.1 Does your country have its own autograter (online judge)?	YES	NO
If YES which is the authority responsible for it :		
If YES who can have access to it? (Choose one of the following answewrs)		
1. Only students and trainers from my country		
2. Everyone		
3. Other (Please complete):		

2.2 Is it important for a country to posses its own autograter (online judge) for the preparation of its students?			
1. No impact	2. Minimal	3. Moderate	4. Great

2.3 Is it possible to achieve the same results by using a third party autograter (online judge)	YES	NO
If NO please explain :		

2.4 What are the main benefits for the students when they using an online judge system for training? ***(You can choose multiple answers)***
1. They are getting familiar with the IOI contest environment
2. They can progress faster since they do not need someone to check their solutions
3. They find it as game because of the point system and/or the high score table
4. Their solutions must past several test cases
5. Other (Please complete):

2.5 What are the main problems for the students when they starting using an online judge system? ***(You can choose multiple answers)***
1. No problems at all
2. Using the interface
3. The problems are too difficult for a starter
4. Lack of feedback
5. Other (Please complete):

2.6 Name 3 online judge systems that you would recommend for training
1.
2.
3.

2.7 Which are the factors for your selection above (2.6)
1. Friendly interface
2. Plurality and variety of tasks
3. Training approach (e.g. there is a theory page before at the start of every section)
4. Language selection
5. Other(Please complete):

Section 3: System Functions & Task Structure

3.1 All the tasks must be shown at once?	YES	NO

3.2 Inputs must be available for failed test cases?	YES	NO

3.3 After every correct solution there must be a solution provided by the administrator.	YES	NO

3.4 Before every group of tasks there must be a brief theory?	YES	NO

3.5 All the test cases must score the same points?	YES	NO

3.6 There must be tests at every major section (e.g. Tasks for limited time)?	YES	NO

3.7 There must be a distinction between senior and junior problems?	YES	NO

3.8 Other suggestions

Παράρτημα Β
Συνεντεύξεις

Οι συνεντεύξεις πραγματοποιήθηκαν κατά τη διάρκεια της φάσης της προκαταρκτική έρευνας. Τα άτομα από τα οποία πήρα συνέντευξη ήταν ο Γιώργος Μιχαήλ και ο Μιχάλης Ψάλιος.

Β.1 Συνέντευξη Γιώργου Μιχαήλ – Μιχάλη Ψάλιου

Ο Γιώργος Μιχαήλ ήταν για χρόνια η ψυχή και ο νους της Παγκύπριας Ολυμπιάδας Πληροφορικής. Σαν μαθητής έλαβε μέρος σε πολλούς παγκόσμιους διαγωνισμούς. Στη συνέχεια παρά το νεαρό της ηλικίας του ανέλαβε τη θεματοθέτηση της Παγκύπριας Ολυμπιάδας Πληροφορικής. Επίσης συνόδεψε πολλές φορές την ομάδα μας στο εξωτερικό.

Ο Μιχάλης Ψάλιος είναι μαθητής (στρατιώτης πλέον) και ασχολείται με το διαγωνισμό από την Α Λυκείου. Ήταν μέρος στην ομάδα που εκπροσώπησε την Κύπρο στους διεθνείς διαγωνισμούς το 2012 και το 2013 (η συνέντευξη έγινε κατά τη διάρκεια της βαλκανιάδας το 2012). Το 2013 κέρδισε μάλιστα χάλκινο μετάλλιο στη βαλκανιάδα πληροφορικής.

Π: Πιστεύετε ότι είναι σημαντικό να έχουμε σαν Κύπρος το δικό μας σύστημα αξιολόγησης;

Γ: Πιστεύω ότι είναι πολύ σημαντικό ώστε να ελέγχουμε εμείς την εκπαίδευση των μαθητών μας. Τα συστήματα που υπάρχουν είναι βασισμένα στις ανάγκες άλλων χωρών, πιο προηγμένων από εμάς.

Μ: Νομίζω ότι είναι σημαντικό στην αρχή. Όταν ξεκινάς να ασχολείσαι είναι πολύ εύκολο να απογοητευτείς και να τα παρατήσεις γιατί οι ασκήσεις στα συστήματα όπως το USACO είναι αρκετά δύσκολες.

Π: Ποια συστήματα προτιμάτε;

Γ: Ως μαθητής χρησιμοποιούσα πολύ το USACO. Δεν είναι τέλειο αλλά το βρίσκω πολύ χρήσιμο. Φυσικά πρέπει να είσαι ενός επιπέδου και πάνω. Ένα άλλο σύστημα που μου αρέσει είναι SPOJ.

Μ: Θα συμφωνήσω για το USACO. Εγώ ξεκίνησα με το Hellenico βασικά λόγω γλώσσας. Τώρα άρχισα να ασχολούμαι με το Codeforce και το CodeShef.

Π: Ποια νομίζεται ότι είναι τα πλεονεκτήματα από τη χρήση ενός τέτοιου συστήματος;

Γ: Πάρα πολλά. Όπως θυμάσαι στην Παγκύπρια Ολυμπιάδα Πληροφορικής αρχίσαμε να χρησιμοποιούμε αυτόματο σύστημα αξιολόγησης στους διαγωνισμούς για πρώτη φορά φέτος. Δανεικό μάλιστα αφού χρησιμοποιήσαμε το Hellenico. Το βασικότερο είναι ότι οι μαθητές μας έρχονται σε επαφή με το περιβάλλον που θα αντιμετωπίσουν στους διαγωνισμούς.

Μ: Θα μιλήσω από την εμπειρία μου. Τον πρώτο χρόνο δεν είχα ασχοληθεί πολύ με κανένα σύστημα. Είχα καταφέρει να περάσω μόνο τις 4 πρώτες ασκήσεις στο Hellenico. Φέτος το έχω σχεδόν τελειώσει (το Hellenico έχει περίπου 60 προβλήματα) και νοιώθω πολύ πιο δυνατός.

Π: Να περάσω σε λίγο περισσότερο τεχνικές ερωτήσεις. Πιστεύεται ότι τα προβλήματα πρέπει να εμφανίζονται όλα ταυτόχρονα ή σταδιακά.

Γ: Νομίζω είναι καλύτερα σταδιακά για να υπάρχει μια πιο σωστή πορεία στην εκμάθηση.

Μ: Εγώ θα προτιμούσα να ήταν όλα μαζί. Έτσι θα μπορούσα να δουλέψω με αυτά που με ενδιαφέρουν.

Π: Τα test cases για κάθε άσκηση πρέπει να είναι ορατά από τους μαθητές.

Γ: Νομίζω πως όχι. Αν είμαστε ότι οι μαθητές μας θα αυτοπειθαρχήσουν και θα βασίζουν τις λύσεις στους στα test cases τότε μπορεί να ήταν χρήσιμο.

Μ: Μερικές φορές και εγώ πιστεύω ότι είναι χρήσιμο. Μπορεί να κολλήσεις σε κάποιο test case και να μην καταλαβαίνεις που είναι το λάθος. Συμμερίζομαι όμως την ανησυχία του Γιώργου.

Π: Είναι χρήσιμο να υπάρχει θεωρία;

Γ: Γενικά είναι χρήσιμο όμως και να μην υπάρχει, το internet είναι γεμάτο με πληροφορίες.

Μ: Είναι χρήσιμο όταν ξεκινάς. Όσο προχωράς είναι βασικό να μην έχεις μόνο μια πηγή πληροφόρησης.

Π: Βαθμολογία να υπάρχει;

Γ: Δεν είναι απαραίτητο, αλλά βοηθά στο συναγωνισμό.

Μ: Ναι να φαίνεται ποιος είναι ο καλύτερος. Εγώ δηλαδή (Γέλια)