ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΕΠΛ131 Αρχές Προγραμματισμού Ι

Ακαδημαϊκό Έτος 2015/16 – Εαρινό Εξάμηνο

ΤΕΛΙΚΗ ΕΞΕΤΑΣΗ ΕΞΑΜΗΝΟΥ

ΗΜΕΡΟΜΗΝΙΑ: 9 Μαΐου 2016
ΔΙΑΡΚΕΙΑ: 8:30πμ – 11:00μμ
ΑΙΘΟΥΣΑ: Κτήριο ΘΕΕ, Αίθουσα 147
ΔΙΔΑΣΚΟΥΣΑ: Ελπίδα Κεραυνού-Παπαηλιού

Απαντήστε <u>όλες</u> τις ερωτήσεις
Κάθε ερώτηση βαθμολογείται με 25 μονάδες

Καλή επιτυχία!

Ερώτηση 1

Ολοκληρώστε το ακόλουθο πρόγραμμα μέσω των ορισμών των σχετικών συναρτήσεων. Το πρόγραμμα υπολογίζει και παρουσιάζει τον αριθμό εμφανίσεων, δηλαδή τη συχνότητα, κάθε γραμματικού χαρακτήρα (όπου τα κεφαλαία γράμματα δεν διακρίνονται από τα μικρά γράμματα) σε κάποιο αρχείο κειμένου, το οποίο προσδιορίζεται μέσω επανακατεύθυνσης όπως υποδεικνύεται στο παράδειγμα που δίνεται στο τέλος της ερώτησης.

```
public class Frequencies {

public static void putchar(char c){
     System.out.print(c);}
```

/* Να οριστεί η συνάρτηση **low_case** η οποία λαμβάνει ένα χαρακτήρα. Αν ο χαρακτήρας αυτός είναι κεφαλαίο γράμμα (‘A’ … ‘Z’), επιστρέφει το αντίστοιχο μικρό γράμμα, διαφορετικά επιστρέφει τον αρχικό χαρακτήρα. */

```
public static char low_case (char c){ . . . }
```

/* Να οριστεί η συνάρτηση **initialize** η οποία αρχικοποιεί όλες τις εισόδους του πίνακα Table στην τιμή val. */

```
public static void initialize (int [] Table, int val){
 . . . .
}
```

/* Να οριστεί η συνάρτηση **draw_line** η οποία εκτυπώνει το χαρακτήρα c, len φορές */

```
public static void draw_line (int len, char c){ .  . . }
```

/* Να οριστεί η συνάρτηση **count_freqs**, η οποία μετρά και καταχωρεί στον πίνακα Table τις συχνότητες, μόνο των γραμματικών χαρακτήρων σε κάποιο αρχείο (το αρχείο προσδιορίζεται μέσω επανακατεύθυνσης όταν εκτελείται το πρόγραμμα). Οι συχνότητες των μικρών και αντίστοιχων κεφαλαίων γραμμάτων μετρούνται μαζί. (Υπόδειξη: Η συνάρτηση να κάνει χρήση των συναρτήσεων initialize_table και low_case.) */

```
public static void count_freqs(int [] Table){ . . . }
```

/* Να οριστεί η συνάρτηση **display_freqs** η οποία παρουσιάζει τα περιεχόμενα του πίνακα Table, όπως δεικνύεται στο πιο κάτω παράδειγμα:

Έστω ότι ο πίνακας Freqs ορίζεται και αρχικοποιείται ως εξής:

```
int [] Freqs = {51,7,67,12,57,41,1,13,60,0,0,27,7,
                53,20,12,16,53,24,49,12,6,7,5,4,11};
```

Η κλήση display_table(Freqs) παρουσιάζει τα περιεχόμενα του πίνακα Freqs ως ακολούθως:

```
Letter Frequencies

a   51  b    7  c   67  d   12  e   57
f   41  g    1  h   13  i   60  j    0
k    0  l   27  m    7  n   53  o   20
p   12  q   16  r   53  s   24  t   49
u   12  v    6  w    7  x    5  y    4
z   11
```

Συνεπώς, να θεωρηθεί ότι οι είσοδοι του πίνακα αντιστοιχούν στα γράμματα του αλφαβήτου, και τα περιεχόμενα των εισόδων δίνουν τις συχνότητες των εν λόγω γραμματικών χαρακτήρων. Οι συχνότητες παρουσιάζονται από τη συνάρτηση display_table ανά πέντε γράμματα σε κάθε σειρά. */

```
public static void display_freqs(int [] Table){ . . . }
```

/* Να οριστεί η συνάρτηση **draw_histogram** η οποία παρουσιάζει τον πίνακα συχνοτήτων Table σε μορφή ιστογράμματος όπως δεικνύεται στο παράδειγμα που δίνεται στο τέλος της ερώτησης. Η συνάρτηση παρουσιάζει επίσης τα γράμματα που έχουν τη μέγιστη συχνότητα, καθώς και τα γράμματα που έχουν μηδενική συχνότητα. (Υπόδειξη: Η συνάρτηση να κάνει χρήση της συνάρτησης draw_line, όπου κάθε αστεράκι ('*') αντιστοιχεί σε δύο εμφανίσεις του δεδομένου γράμματος. */

```
public static void draw_histogram(int [] Freqs){ . . . . }
```

/* Η κυρίως συνάρτηση ορίζεται ως ακολούθως: */

```
public static void main (String [] args) {
 int [] Freqs = new int[26];
 count_freqs(Freqs);
 display_freqs(Freqs);
 draw_histogram(Freqs);
 putchar('\n'); putchar('\n');
}

}
```

Ακολουθεί παράδειγμα χρήσης της προγράμματος όπου ifile.txt είναι κάποιο αρχείο κειμένου:

```
$ java Frequencies < ifile.txt


Letter Frequencies

a   73     b   17     c   92     d   17     e   69
f   42     g    5     h   20     i   91     j    0
k    0     l   38     m   19     n   63     o   32
p   27     q   19     r   68     s   61     t   98
u   33     v    9     w   10     x    5     y   14
z   11

Histogram of Letter Frequencies

a     ************************************
b     ********
c     *********************************************
d     ********
e     ***********************************
f     ********************
g     **
h     **********
i     *********************************************
j
k
l     ******************
m     *********
n     ******************************
```

```
o     ****************
p     *************
q     *********
r     ********************************
s     ****************************
t     ************************************************
u     ****************
v     ****
w     *****
x     **
y     *******
z     *****

Letters with max frequency:  t

Letters with zero frequency:  j   k
```

Ερώτηση 2

Αναπτύξετε πρόγραμμα το οποίο διαβάζει τις επιδόσεις ενός συνόλου μαθητών-αθλητών, στα 100, 200, και 400 μέτρα τρέξιμο, και για κάθε άθλημα παρουσιάζει την κατάταξη των επιδόσεων των αθλητών από την καλύτερη στη χειρότερη. Για παράδειγμα, αν θεωρήσουμε ότι οι μαθητές-αθλητές είναι 10 και οι επιδόσεις τους καταγράφονται στο αρχείο `AthletesData.txt` ως ακολούθως (η πρώτη στήλη δίνει τον αριθμό του αθλητή και οι επόμενες τρεις στήλες δίνουν τις επιδόσεις του, σε αριθμό δευτερολέπτων, στα 100, 200 και 400 μέτρα τρέξιμο αντίστοιχα)

```
1       10      22      40
2       17      25      47
3       11      19      39
4       14      20      41
5       18      25      42
6       12      24      39
7       10      20      30
8       13      24      46
9       14      26      47
10      10      19      32
```

η πιο κάτω εντολή εκτέλεσης του προγράμματος

```
$ java Athletes < AthletesData.txt
```

δημιουργεί τα ακόλουθα αποτελέσματα:

```
Athletes Performance in 100 m
-----------------------------
Athlete 1 has performance 10 seconds
Athlete 7 has performance 10 seconds
Athlete 10 has performance 10 seconds
Athlete 3 has performance 11 seconds
Athlete 6 has performance 12 seconds
Athlete 8 has performance 13 seconds
Athlete 4 has performance 14 seconds
Athlete 9 has performance 14 seconds
```

```
Athlete 2 has performance 17 seconds
Athlete 5 has performance 18 seconds

Athletes Performance in 200 m
-----------------------------
Athlete 3 has performance 19 seconds
Athlete 10 has performance 19 seconds
Athlete 4 has performance 20 seconds
Athlete 7 has performance 20 seconds
Athlete 1 has performance 22 seconds
Athlete 6 has performance 24 seconds
Athlete 8 has performance 24 seconds
Athlete 2 has performance 25 seconds
Athlete 5 has performance 25 seconds
Athlete 9 has performance 26 seconds

Athletes Performance in 400 m
-----------------------------
Athlete 7 has performance 30 seconds
Athlete 10 has performance 32 seconds
Athlete 3 has performance 39 seconds
Athlete 6 has performance 39 seconds
Athlete 1 has performance 40 seconds
Athlete 4 has performance 41 seconds
Athlete 5 has performance 42 seconds
Athlete 8 has performance 46 seconds
Athlete 2 has performance 47 seconds
Athlete 9 has performance 47 seconds
```

Για την απάντησή σας μπορείτε να συμπληρώσετε τον ακόλουθο σκελετό προγράμματος:

```
public class Athletes{

 public static final int NUM_ATHLETES = 10; /* ο αριθμός
                                                των αθλητών */
```

/* Η συνάρτηση **swap** εναλλάσσει τα περιεχόμενα των αντίστοιχων θυρίδων των δύο μονοδιάστατων πινάκων που λαμβάνει ως παραμέτρους */

```
public static void swap (int [] athlete_1,
                         int [] athlete_2){ . . . }
```

/* Η συνάρτηση **min_pos** επιστρέφει την είσοδο του τμήματος του πίνακα Table, που προσδιορίζεται από τις σειρές start_row και end_row, η οποία περιέχει το μικρότερο στοιχείο στη στήλη col – σε περίπτωση που υπάρχουν πέραν της μίας εισόδου στο εν λόγω τμήμα του πίνακα που περιέχουν το ίδιο μικρότερο στοιχείο στη στήλη col, τότε από αυτές επιστρέφεται η είσοδος με την μικρότερη τιμή στην πρώτη στήλη */

```
public static int min_pos (int [][] Table, int start_row,
                           int end_row, int col){ . . }
```

/* Η συνάρτηση **select_sort** ταξινομεί το αρχικό τμήμα, μήκους len, του πίνακα Athletes σε αύξουσα σειρά ως προς τα περιεχόμενα της στήλης col του πίνακα,

χρησιμοποιώντας τις συναρτήσεις min_pos και swap – συγκεκριμένα η συνάρτηση εφαρμόζει τον ακόλουθο αλγόριθμο: αρχίζοντας από ολόκληρο το υπό εξέταση τμήμα του πίνακα, και διαδοχικά μειώνοντας το κατά ένα στοιχείο «αποκόβοντας» το πρώτο του στοιχείο, μέχρι να υπάρχει μόνο ένα στοιχείο, εντοπίζει το μικρότερο σε σειρά στοιχείο του τμήματος και το εναλλάσσει με το πρώτο στοιχείο του τμήματος */

```
public static void select_sort (int [][]Athletes,
                                int col, int len)
{. . . .}
```

/* Η συνάρτηση get_data εισαγάγει τα στοιχεία για num μαθητές-αθλητές στον πίνακα Athletes */

```
public static void get_data (int [][]Athletes, int num)
 { . . . }
```

/* Η συνάρτηση display_data παρουσιάζει τα στοιχεία από τον πίνακα Athletes για τους πρώτους rows αθλητές, σε σχέση με τη στήλη col – βλέπε πιο πάνω παράδειγμα */

```
public static void display_data (int [][]Athletes,
                                 int rows, int col)
  { . . . }
```

/* Η συνάρτηση main ορίζεται ως ακολούθως */

```
public static void main (String[] args){
 int [][]Athletes_Data = new int [NUM_ATHLETES][4];
 get_data(Athletes_Data, NUM_ATHLETES);
 for (int i=1; i <= 3; i++) {
   select_sort(Athletes_Data,i,NUM_ATHLETES);
   display_data(Athletes_Data, NUM_ATHLETES,i);
  } System.out.println(); System.out.println();
 }

}
```

Ερώτηση 3

Η κλάση αντικειμένων CSet αντιπροσωπεύει σύνολα από τους 26 γραμματικούς χαρακτήρες {a, b, c, …, x, y, z}. Πιο κάτω δίνεται ο σκελετός της κλάσης για τον εν λόγω νέο τύπο δεδομένων, ο οποίος χρειάζεται να ολοκληρωθεί με τους ορισμούς των μεθόδων αναφοράς:

```
public class CSet {

 protected static final int ALPHABET = 26;
```

/* Το πεδίο size δίνει το πλήθος των στοιχείων του υπό αναφορά συνόλου */

```
 private int size;
```

/* Το πεδίο elements δίνει τα στοιχεία του υπό αναφορά συνόλου. Είναι ένας πίνακας, οι είσοδοι του οποίου αντιστοιχούν στους 26 γραμματικούς χαρακτήρες a, b, .., z. Οι χαρακτήρες που περιλαμβάνονται στο σύνολο επισημαίνονται με true στις

σχετικές εισόδους. Συνεπώς το σύνολο δεν μπορεί να περιλαμβάνει τον ίδιο χαρακτήρα πέραν της μιας φοράς */

```
private boolean[] elements = new boolean[ALPHABET];
```

/* Να οριστούν οι ακόλουθες μέθοδοι αναφοράς */

/* Ο πιο κάτω (πρώτος) κατασκευαστής δημιουργεί κενό ή πλήρες (καθολικό) σύνολο σύμφωνα με την τιμή της παραμέτρου empty */

```
public CSet (boolean empty){ . . . . }
```

/* Ο πιο κάτω (δεύτερος) κατασκευαστής δημιουργεί σύνολο από τον κάθε ξεχωριστό γραμματικό χαρακτήρα, και μόνο, που περιλαμβάνεται στη γραμματοσειρά cs, όπου τυχόν κεφαλαία γράμματα μετατρέπονται στα αντίστοιχα μικρά. */

```
public CSet (String cs){ . . . . }
```

/* Ο πιο κάτω (τρίτος) κατασκευαστής δημιουργεί σύνολο αντιγράφοντας το σύνολο s (copy constructor). **Να οριστεί βάσει του δεύτερου κατασκευαστή.** */

```
public CSet (CSet s) {. . . .}
```

/* Επιστρέφει το πλήθος των στοιχείων του υπό αναφορά συνόλου */

```
public int Size (){ . . . .}
```

/* Μετατρέπει το υπό αναφορά σύνολο σε εκτυπώσιμη μορφή, δηλαδή την ακολουθία, σε αλφαβητική σειρά, των γραμματικών χαρακτήρων που το απαρτίζουν */

```
public String toString(){. . . . }
```

/* Προσθέτει το χαρακτήρα c στο υπό αναφορά σύνολο, νοουμένου ότι είναι μικρό ή κεφαλαίο γράμμα */

```
public void insertMember (char c){ . . . . }
```

/* Διαγράφει το χαρακτήρα c από το υπό αναφορά σύνολο, νοουμένου ότι ο c ανήκει στο σύνολο */

```
public void deleteMember (char c){ . . . .  }
```

/* Ανήκει ο χαρακτήρας c στο υπό αναφορά σύνολο; */

```
public boolean isMember (char c){ . . . . }
```

/* Περιλαμβάνει το υπό αναφορά σύνολο τα ίδια στοιχεία με το σύνολο cs**; Ο ορισμός να μην κάνει χρήση του πεδίου elements.** */

```
public boolean equal (CSet cs){ . . . . }
```

/* Είναι το σύνολο cs υποσύνολο του υπό αναφορά συνόλου; **Ο ορισμός να μην κάνει χρήση του πεδίου elements.** */

```
 public boolean subset (CSet cs){ . . . . }
```

/* Επιστρέφει το σύνολο που είναι η ένωση του υπό αναφορά συνόλου με το σύνολο cs. **Ο ορισμός να μην κάνει χρήση του πεδίου elements.** */

```
 public CSet union (CSet cs){ . . . . }
```

/* Επιστρέφει το σύνολο που είναι η τομή του υπό αναφορά συνόλου με το σύνολο cs. */

```
 public CSet intersection (CSet cs){ . . . . }
```

/* Επιστρέφει το σύνολο που είναι η διαφορά του υπό αναφορά συνόλου με το σύνολο cs. */

```
 public CSet difference (CSet cs){ . . . . }
```

/* Επιστρέφει το σύνολο που είναι η διαφορά του καθολικού συνόλου από το υπό αναφορά σύνολο */

```
 public CSet complement (){ . . . . }

}
```

Ερώτηση 4

Απαντήστε είτε το Μέρος (i), είτε το Μέρος (ii) της ερώτησης 4 (κάθε Μέρος βαθμολογείται με 25 μονάδες). Αν απαντήσετε και τα δύο Μέρη, τότε ο βαθμός από τις επιπρόσθετες 25 μονάδες θα προσμετρηθεί ως bonus.

Μέρος (i)

Ένας **αναγραμματισμός** είναι μία λέξη που σχηματίζεται με αναδιάταξη των γραμμάτων κάποιας άλλης λέξης. Για παράδειγμα, η λέξη `carthorse` είναι αναγραμματισμός της λέξης `orchestra` και αντίστροφα.

Αναπτύξετε πρόγραμμα το οποίο διαβάζει (από το τρέχον κανάλι εισόδου) ένα αριθμό λέξεων (με μέγιστο αριθμό τις 100 λέξεις) και στη συνέχεια εκτυπώνει το κάθε ζεύγος διαφορετικών λέξεων που αποτελούν αναγραμματισμούς μεταξύ τους. Για παράδειγμα, αν το αρχείο **`words.txt`** περιλαμβάνει τις λέξεις

```
carthorse
horse
horsecart
idonotknowu
okinowdonut
orchestra
```

τότε η εκτέλεση του προγράμματος είναι η ακόλουθη:

```
% java Anagram < words.txt
```

```
carthorse = horsecart
carthorse = orchestra
horsecart = orchestra
idonotknowu = okinowdonut
```

Για την απάντησή σας μπορείτε να συμπληρώσετε τον ακόλουθο σκελετό προγράμματος:

```
public class Anagram {

 public static final int WORDNUM = 100;

 /* Πόσες φορές εμφανίζεται ο χαρακτήρας c στη λέξη s; */

 public static int occurs (String s, char c){ . . . . }

/* Είναι οι δύο λέξεις w1 και w2 η μια αναγραμματισμός της άλλης; */

 public static boolean anagram(String w1, String w2){ . . . }

/* Εισαγωγή λέξεων και επιστροφή του πλήθους τους */

 public static int ReadWords(String[] Words){ . . . . }

 /* Ολοκλήρωση προγράμματος */

 public static void main (String [] args){
     . . . .
 }
}
```

Μέρος (ii)

(α) Πιο κάτω ορίζεται η κλάση Prog, η οποία περιλαμβάνει τον ορισμό της συνάρτησης Fn. Εξηγήστε τη λειτουργία της Fn, παράλληλα κάνοντας αναφορά σε στοιχεία της γλώσσας Java, καθώς και στις αλγοριθμικές δομές, που διαφαίνονται μέσω του ορισμού της συνάρτησης αυτής.

```
public class Prog{

public static String[] Fn (String[] as, String[] bs){
  int N = as.length + bs.length;
  String [] rs = new String[N];
  for (int i = 0; i < as.length; i++)
         rs[i] = new String(as[i]);
  for (int i = as.length; i < N; i++)
         rs[i] = new String(bs[i - as.length]);
  return rs;
 }

 public static String[] Fn (String s){
   if (s.length() == 0) {
       String[] xs = new String[1];
       xs[0] = new String("");
       return xs;}
```

```
   else {
      String [] xs = Fn(s.substring(1, s.length()));
      String [] ys = new String[xs.length];
      for (int i = 0; i < ys.length; i++){
           ys[i] = new String(xs[i]);
           ys[i] = ys[i].concat(s.substring(0,1));}
      return Fn(xs,ys);
   }
 }

 public static void main (String[] args){
   String [] ss = Fn(args[0]);
   for (int i = 0; i < ss.length; i++)
       System.out.println("ss-" + i + " = " + ss[i]);
 }
}
```

Με δεδομένο τον πιο πάνω ορισμό της κλάσης Prog, ποιο θα είναι το αποτέλεσμα της εκτέλεσης της εντολής

```
$ java Prog ABC
```

από τις ακόλουθες τρεις; Τεκμηριώστε την απάντησή σας, σύμφωνα και με την επεξήγηση που έχετε δώσει για τη συνάρτηση Fn.

Απάντηση 1.	Απάντηση 2.	Απάντηση 3.
ss-0 = ABC ss-1 = ACB ss-2 = BAC ss-3 = BAC ss-4 = BCA ss-5 = CAB ss-6 = CBA	ss-0 = ss-1 = C ss-2 = B ss-3 = CB ss-4 = A ss-5 = CA ss-6 = BA ss-7 = CBA	ss-0 = ss-1 = A ss-2 = B ss-3 = C ss-4 = AB ss-5 = AC ss-6 = BCA ss-7 = ABC

(β) Πιο κάτω ορίζετε η κλάση Unknown. Εξηγήστε τη λειτουργία της.

```
public class Unknown {

  public static void Rn (double s, int N){
    if (StdIn.isEmpty())
         System.out.println("Result = " + (s/N));
    else {
       double x = StdIn.readDouble();
       Rn (s+x, ++N);
       System.out.println(x);
    }
  }

  public static void main(String[] args){
      Rn(0.0, 0);
  }
}
```

Έστω ότι το αρχείο κειμένου `Data.txt` περιλαμβάνει τους ακόλουθους αριθμούς:

```
34 25 67 90 10
```

Ποιο θα είναι το αποτέλεσμα της εκτέλεσης της εντολής

```
$ java Unknown < Data.txt
```

από τις ακόλουθες τρεις; Τεκμηριώστε την απάντησή σας.

Απάντηση 1.	Απάντηση 2.	Απάντηση 3.
Result = 45.2 10.0 90.0 67.0 25.0 34.0	10.0 90.0 67.0 25.0 34.0 Result = 45.2	Result = 45.2 34.0 25.0 67.0 90.0 10.0

Θα υπάρξει αλλαγή ή όχι στη λειτουργία της συνάρτησης `Rn` αν η έκφραση `++N` αντικατασταθεί από την έκφραση `N++` όπως φαίνεται πιο κάτω; Τεκμηριώστε την απάντησή σας.

```
public static void Rn (double s, int N){
  if (StdIn.isEmpty())
       System.out.println("Result = " + (s/N));
  else {
     double x = StdIn.readDouble();
     Rn (s+x, N++);
     System.out.println(x);
   }
}
```

ΤΕΛΟΣ ΕΡΩΤΗΣΕΩΝ

ΧΩΡΟΣ ΑΠΑΝΤΗΣΕΩΝ

Ονοματεπώνυμο Φοιτητή: ----------

Ταυτότητα: ----------

Υπογραφή: ----------

Ερωτήσεις	**Μονάδες**
Ερώτηση 1	
Ερώτηση 2	
Ερώτηση 3	
Ερώτηση 4 [Μέρος (i)]	
Ερώτηση 4 [Μέρος (ii)]	
Σύνολο Μονάδων	

Παρατηρήσεις Διδάσκοντα

Απάντηση στην Ερώτηση 1

(τυχόν συνέχεια στην απάντηση της Ερώτησης 1)

Απάντηση στην Ερώτηση 2

(τυχόν συνέχεια στην απάντηση της Ερώτησης 2)

Απάντηση στην Ερώτηση 3

(τυχόν συνέχεια στην απάντηση της Ερώτησης 3)

Απάντηση στην Ερώτηση 4

(τυχόν συνέχεια στην απάντηση της Ερώτησης 4)

(τυχόν συνέχεια στην απάντηση της Ερώτησης 4)

Επιπρόσθετο φύλλο (για συμπλήρωση απάντησης ή πρόχειρο)

Επιπρόσθετο φύλλο (για συμπλήρωση απάντησης ή πρόχειρο)

Επιπρόσθετο φύλλο (για συμπλήρωση απάντησης ή πρόχειρο)

Επιπρόσθετο φύλλο (για συμπλήρωση απάντησης ή πρόχειρο)

Επιπρόσθετο φύλλο (για συμπλήρωση απάντησης ή πρόχειρο)

ΤΕΛΟΣ ΕΞΕΤΑΣΤΙΚΟΥ ΔΟΚΙΜΙΟΥ