**Τμήμα Μηχανικών Η/Υ & Πληροφορικής**
**Πανεπιστήμιο Πατρών**
**235577 Εξασφάλιση Ποιότητας και Πρότυπα**

# 4η Εργασία

## Διαδικαστικά

Η εργασία είναι **αυστηρά ατομική** και αποτελεί την 4η από τις 5 εργασίες του μαθήματος. Ως 5η εργασία θα υπολογιστεί η συμμετοχή στη διόρθωση μιας εργασίας. Τα διαδικαστικά που αφορούν τις εργασίες αναφέρονται αναλυτικά στις πληροφορίες του μαθήματος στο eClass. **Αντιγραφή σε κάποια εργασία συνεπάγεται μηδενισμό σε όλες τις εργασίες αυτού του έτους**.

**Όλες οι εργασίες θα παραδοθούν αυστηρά μέσω eClass.**

Η 4η εργασία έχει καταληκτική ημερομηνία και ώρα παράδοσης **Κυριακή 9/01/2022** και ώρα **23:30** (πείτε στον εαυτό σας ότι το σύστημα κλείνει 11 το βράδυ και ότι η μισή ώρα είναι για να μην τύχει κάτι). **Καμία εργασία δεν θα γίνει δεκτή μετά τη λήξη της προθεσμίας**[1].

## ΣΗΜΑΝΤΙΚΟ:

Για την εργασία παραδώστε μόνο ένα αρχείο pdf (π.χ. Xenos_Michalis.pdf) με το όνομά σας. Μέσα στο κείμενο δεν θα πρέπει να υπάρχει καμία πληροφορία για εσάς (ούτε όνομα, ούτε αριθμό μητρώου, ούτε τίποτε άλλο). **Όταν μετονομάσουμε το αρχείο σας σε κάτι άλλο θα πρέπει να είναι τελείως ανώνυμα!** Αυτό περιλαμβάνει και τα metadata του αρχείου, δηλαδή **να σβήσετε κάθε προσωπική πληροφορία και από τα properties του αρχείου** (π.χ. όνομα). Υπάρχει κώδικας που το κάνει στο eClass (σε python) και θα βρείτε δεκάδες εργαλεία online. Σε περίπτωση που η εργασία σας δεν είναι ανώνυμη θα διορθωθεί και θα βαθμολογηθεί κανονικά, **αλλά θα λάβει -30% του βαθμού ως ποινή**. Είναι κρίμα να χάνετε μονάδες έτσι άρα ελέγξτε τα αρχεία σας!

[1] Αυτό είναι κάτι που το τηρώ αυστηρά και δεν θα παρεκκλίνω ποτέ, άρα μην στείλετε εργασία έστω και 1 λεπτό μετά τη λήξη της προθεσμίας με e-mail.

## Ζητούμενο 1 (μονάδες 6)

Δίνεται το παρακάτω πρόγραμμα σε C. Δεν σας ενδιαφέρει τι κάνει (πρακτικά έχουμε “πειράξει” κάποιο κώδικα για τις ανάγκες του ελέγχου).

```
#include <stdio.h>
int main() {
  int year, n, rev=0, rem;
  printf("Enter a year: ");
  scanf("%d", &year);

  // leap year if perfectly divisible by 400
  if (year % 400 == 0) {
    printf("%d is a leap year.\n", year);
    n=-1;
  }
  // not a leap year if divisible by 100 but not divisible by 400
  else if (year % 100 == 0) {
    printf("%d is not a leap year.\n", year);
    n=1;
  }
  // leap year if not divisible by 100 but divisible by 4
  else if (year % 4 == 0) {
    printf("%d is a leap year.\n", year);
    n=-1;
  }
  // all other years are not leap years
  else {
    printf("%d is not a leap year.\n", year);
    n=12;
  }

   // true if year is perfectly divisible by 2
   if(year % 2 == 0)
   {
      printf("%d is even.\n",year);
   }
   else
   {
      printf("%d is odd.\n", year);
      n=1234;

   }

 while (n != 0) {
      rem = n % 10;
      rev = rev * 10 + rem;
      n /= 10;
   }
   printf("Reversed number = %d", rev);

if(n<-1)
{
    n = -2;
    while(n<0);
    n = n + 10;
}
   return 0;
}
```

Για το παραπάνω πρόγραμμα:

**1)** Να κατασκευαστεί ο γράφος της κυκλωματικής πολυπλοκότητας και να υπολογιστεί η κυκλωματική πολυπλοκότητα και με τους 3 τρόπους που έχετε διδαχθεί. Για να είναι σαφής και κατανοητός ο γράφος της κυκλωματικής πολυπλοκότητας που θα σχεδιάσετε, θα πρέπει να αναφέρετε για κάθε κόμβο του γράφου σε ποιες εντολές αντιστοιχεί. Απάντηση που θα έχει μόνο το γράφο, χωρίς επεξήγηση και αρίθμηση, δεν θα θεωρείται ολοκληρωμένη.

**2)** Να καταγραφούν τα βασικά μονοπάτια. Για τη δική σας διευκόλυνση λύστε την άσκηση ως εξής: α) Βρείτε τις εξαρτήσεις συνύπαρξης E1, E2, … En που υπάρχουν στο γράφο σας. β) Με βάση αυτές τις εξαρτήσεις βρείτε το μικρότερο δυνατό μονοπάτι που να είναι έγκυρο, ονομάστε το M1 και εμπλουτίστε το με όσο το δυνατό λιγότερες νέες ακμές είναι εφικτό, ξεκινώντας από τον πρώτο κόμβο όπου υπάρχει δυνατότητα διακλάδωσης. Αν το νέο μονοπάτι είναι έγκυρο ονομάστε το M2 και επαναλάβετε (στο M3, … Mn), αλλιώς αναζητήσετε άλλο κόμβο διακλάδωσης. γ) Συνεχίστε μέχρι είτε να μην υπάρχουν νέες ακμές, είτε να μην υπάρχουν έγκυρα μονοπάτια που να περιέχουν όλες τις ακμές.

**3)** Να σχεδιαστούν οι περιπτώσεις ελέγχου με βάση την τεχνική δοκιμής βασικών μονοπατιών εκτέλεσης.

**Ακολουθήστε το υπόδειγμα λύσης που βρίσκεται στο τέλος της εκφώνησης.**

## Ζητούμενο 2 (μονάδες 4)

Δίνεται το παρακάτω πρόγραμμα σε Java, το οποίο διαβάζει από ένα αρχείο ακεραίους και εξετάζει το αν είναι τέλειοι αριθμοί ή όχι.

Τέλειος λέγεται ένας φυσικός αριθμός όταν το άθροισμα των διαιρετών του, εκτός του ίδιου του αριθμού, ισούται με τον αριθμό. Ο μικρότερος τέλειος αριθμός είναι ο 6. Οι διαιρέτες του 6 είναι οι 1, 2, 3 και το άθροισμα αυτών είναι ίσο με 6 (1+2+3=6). Άλλοι τέλειοι αριθμοί είναι οι 28 = 1 + 2 + 4 + 7 + 14, 496 = 1 + 2 + 4 + 8 + 16 + 31 + 62 + 124 + 248 και ο 8128. Αυτοί ήταν και οι πρώτοι τέλειοι αριθμοί που ανακάλυψε ο Ευκλείδης.

Σημειώσεις για τον τρόπο λειτουργίας του προγράμματος (αν και δεν χρειάζεται να το τρέξετε):

1. *Το αρχείο μπορεί να βρίσκεται σε όποιο φάκελο επιθυμείτε, απλά αλλάξτε το σχετικό μονοπάτι. Αν δεν υπάρχει το αρχείο, τότε σταματάει η εκτέλεση του προγράμματος εμφανίζοντας ένα κατάλληλο μήνυμα.*
2. *Σε κάθε γραμμή του αρχείου πρέπει να βρίσκεται μόνο ένας ακέραιος αριθμός (επιτρέπονται κενά ή tabs στην αρχή ή στο τέλος της γραμμής).*
3. *Αν από μία γραμμή δεν μπορεί να εξαχθεί ένας έγκυρος ακέραιος αριθμός (π.χ. υπάρχουν γράμματα, υπάρχουν 2 ακέραιοι, είναι άδεια η γραμμή, κοκ), τότε σταματάει η εκτέλεση του προγράμματος εμφανίζοντας ένα κατάλληλο μήνυμα.*

```
import java.io.*;
import java.util.Scanner;

public class EAP_PLI42_GE5 {
   public static void main(String[] args) {
     String fileName="C:\\numbers.txt";
     String line= "";
     int number=0;
            try {
        Scanner s = new Scanner(new File(fileName));
        while (s.hasNextLine())
        {
           line = s.nextLine().trim();
           number = Integer.parseInt(line);
           int temp = 0;
           for(int i=1;i<=number/2;i++)
              if(number%i == 0)
                 temp += i;
           if(temp == number)
              System.out.println(number + " is a perfect number");
           else
              System.out.println(number + " is NOT a perfect number");
        }
     } catch (FileNotFoundException ex) {
           System.out.println(fileName+": File not Found!");
     } catch (NumberFormatException ex) {
           System.out.println("'"+line+"' is not a number !");
     }
   }
}
```

Για το παραπάνω πρόγραμμα:

1. Προχωρήστε σε αρίθμηση του κώδικα (αν θέλετε μπορείτε να τον κάνετε c&p στη λύση σας και να προσθέσετε κενές γραμμές για λόγους αναγνωσιμότητας).
2. Να κατασκευαστεί ο γράφος της κυκλωματικής πολυπλοκότητας και να υπολογιστεί η κυκλωματική πολυπλοκότητα και με τους 3 τρόπους που γνωρίζετε.
3. Να καταγραφούν τα βασικά μονοπάτια.
4. Να σχεδιαστούν οι περιπτώσεις ελέγχου με βάση την τεχνική δοκιμής βασικών μονοπατιών εκτέλεσης.

**Σημείωση:** *Για να είναι σαφής και κατανοητός ο γράφος της κυκλωματικής πολυπλοκότητας που θα σχεδιάσετε, θα πρέπει να αναφέρετε για κάθε κόμβο του γράφου σε ποιες εντολές αντιστοιχεί. Απάντηση που θα έχει μόνο το γράφο, χωρίς επεξήγηση και αρίθμηση, δεν θα θεωρείται ολοκληρωμένη.*

**Ακολουθήστε το υπόδειγμα λύσης που βρίσκεται στο τέλος της εκφώνησης.**

---

*Μην αφήνετε την εργασία για τελευταία στιγμή και **ΜΗΝ εμπλακείτε σε διαδικασίες που μπορεί να σας φέρουν σε δύσκολη θέση**.*

**1)** Αρχικά γίνεται η απαρίθμηση των κόμβων.

*(παράδειγμα για το ζητούμενο 1, ακολουθήστε αντίστοιχο για το ζητούμενο 2)*

```
#include <stdio.h>
int main() {
int year, n, rev=0, rem;
printf("Enter a year: ");
scanf("%d", &year);
```

1

Συνεχίστε…

Η αρίθμηση στο 1 είναι σωστή, αλλά προφανώς υπάρχουν και άλλες σωστές. Μπορείτε να ξεκινήσετε από αυτή ή να την αλλάξετε.

Με βάση την παραπάνω αρίθμηση ο γράφος ροής του προγράμματος είναι ο εξής:

Add graph here!

Προσοχή δείξτε και τις περιοχές στο σχήμα σας!

Από τον παραπάνω γράφο προκύπτει ότι η κυκλωματική πολυπλοκότητα είναι add a number, γιατί:

V(g) = e-n+2 = …

Περιοχές γράφου = … (όπως φαίνονται στο σχήμα)

1 + … + … + = …

**2)** Για τον εντοπισμό των βασικών μονοπατιών ακολουθούμε την εξής προσέγγιση:

- Πάρε το μικρότερο δυνατό μονοπάτι (με τις λιγότερες ακμές) το οποίο να είναι έγκυρο.
- Εμπλούτισε αυτό το μονοπάτι **με όσο το δυνατόν λιγότερες νέες ακμές**, ξεκινώντας από τον πρώτο κόμβο που έχεις αυτή τη δυνατότητα διακλάδωσης. Έλεγχος αν αυτό το μονοπάτι είναι έγκυρο, αλλιώς επανέλαβε αυτό το βήμα.
- Συνέχισε μέχρι να μην υπάρχουν νέες ακμές.

Ας δούμε πρώτα όλες τις **εξαρτήσεις συνύπαρξης** που υπάρχουν στον γράφο του λογισμικού μας:

E1. Αν σε ένα μονοπάτι υπάρχει ο κόμβος …
E2. …
E3. …
E4. …
Προσθέστε κι άλλες αν χρειάζονται ή σβήστε!

Με βάση τις παραπάνω εξαρτήσεις το μικρότερο έγκυρο μονοπάτι είναι το εξής:

**M1: 1-2- …**

Στη συνέχεια, ακολουθώντας τον αλγόριθμο έχουμε (με περίγραμμα εμφανίζονται η νέα ή οι νέες ακμές που προστίθενται σε σχέση με τα προηγούμενα βασικά μονοπάτια):

**M2: …**

**M3: 1-2-3-4-… δείξτε κάθε νέα ακμή έτσι**

**M4: …**

**Προσθέστε όσα μονοπάτια χρειάζονται…**

Σε αυτό το σημείο, οι μόνες ακμές που δεν συμπεριλαμβάνονται σε κανένα βασικό μονοπάτι είναι οι ακμές … … (αν υπάρχουν τέτοιες ακμές, μπορεί και όχι). Τυπικά θα έπρεπε να δίναμε μονοπάτια τα οποία θα περιέχουν αυτές τις ακμές, αλλά πρακτικά θα είναι αδύνατο να ελεγχθούν, έτσι δεν χρειάζεται να το κάνουμε. Συνεπώς, το πρόγραμμά μπορεί να ελεγχθεί με προσθέστε νούμερο εδώ βασικά μονοπάτια, δηλαδή λιγότερα από την κυκλωματική πολυπλοκότητα η οποία αποτελεί άνω όριο των βασικών μονοπατιών.

**Σημειώσεις:**

1. *όλες οι ακμές περιλαμβάνονται τουλάχιστον 1 φορά στα παραπάνω μονοπάτια.*
2. *κάθε μονοπάτι διαφέρει από τα άλλα τουλάχιστον σε μία ακμή,*
3. *αυτή είναι μία μόνο από τις ορθές λύσεις (δηλαδή αν ακολουθηθεί διαφορετική μέθοδος καταγραφής των μονοπατιών, το σύνολο των βασικών μονοπατιών μπορεί να είναι διαφορετικό αλλά πάντα θα είναι μεγέθους το νούμερο που είπατε παραπάνω και θα καλύπτει όλες τις ακμές τουλάχιστον 1 φορά).*

**3)** Κάποιες ενδεικτικές περιπτώσεις ελέγχου για τα μονοπάτια είναι:

| Μονοπάτι | Περιγραφή | Περίπτωση ελέγχου (input) | Αναμενόμενο αποτέλεσμα (έξοδος προγράμματος) |
|---|---|---|---|
| M1 | Δίνουμε ως είσοδο | | |
| M2 | | | |
| M3 | | | |
| M4 | | | |
| Προσθέστε όσες γραμμές χρειάζονται, ανάλογα με τα μονοπάτια που είχατε! | | | |