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 παράδοση **06 Ιανουαρίου 2023** 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.

```
#include <stdio.h>

int main()
{
int a, b=0, c=10;

do {
    printf("Give an integer please:");
    scanf("%d", &a);
    } while (a <= 0);

while (b!=a) {
    b++;
    if (c>a) a++;
    else c++;
    }

if (a==b && c!=b) c++;
else c=b;

if (c>50) c=c+a;

printf ("Number is %d \n", c);

return 0;
}
```

Για το παραπάνω πρόγραμμα:

1) Περιγράψτε τι κάνει το πρόγραμμα, δηλαδή δώστε σε απλή γλώσσα τις προδιαγραφές του κώδικα. (Αυτό είναι μια διαδικασία reverse engineering).

2) Επιλέξτε ένα σύνολο 10 τυχαίων τιμών για να επιβεβαιώσετε την ορθότητα της περιγραφής σας και δώστε σε ένα πίνακα την τιμή και το αποτέλεσμα (έξοδος του προγράμματος).

3) Κατασκευάστε το γράφο της κυκλωματικής πολυπλοκότητας και με βάση τον γράφο που σχεδιάσατε 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 αρίθμηση, δεν θα θεωρείται ολοκληρωμένη.

4) 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 Μ1 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 Μ2 και επαναλάβετε (στο Μ3, … Μn), 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.

5) Να σχεδιαστούν οι περιπτώσεις ελέγχου με βάση την τεχνική δοκιμής βασικών μονοπατιών εκτέλεσης δίνοντας 2 τιμές ελέγχου για κάθε μονοπάτι (αν φυσικά υπάρχουν 2 τιμές).

Ακολουθήστε το υπόδειγμα λύσης που βρίσκεται στο τέλος της εκφώνησης και παρουσιάστε την απάντησή σας με παρόμοιο τρόπο

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

Υπόδειγμα λύσης

1) Μια περιγραφή (προφανώς όχι σωστή, απλά για να δείτε τι περίπου θέλουμε) θα μπορούσε να είναι:

Το πρόγραμμα ζητά ως είσοδο ένα αριθμό και αν αυτός είναι μικρότερος του 10 τυπώνει το τετράγωνό του ενώ αν είναι 10 ή μεγαλύτερος του 10 τυπώνει τον ίδιο τον αριθμό.

Κάτι τέτοιο ζητάμε, όχι φιλοσοφική ανάλυση 😊

2) Πίνακας εισόδων εξόδων:

-8 -11	That's great! Give me another number Amazing well done
17	Your number is 188

3) Αρχικά γίνεται η απαρίθμηση των κόμβων.

1

```
#include <stdio.h>


int main()
{
int a, b=0, c=10;
```

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 + … + … + = …

4) 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).*

5) Κάποιες ενδεικτικές περιπτώσεις ελέγχου για τα μονοπάτια είναι:

Μονο πάτι	Περιγραφή	Περίπτωση ελέγχου (input)	Αναμενόμενο αποτέλεσμα (έξοδος προγράμματος)
M1	Δίνουμε ως είσοδο πρώτα -28 και μετά 177	-28 177	Hello world Goodbye cruel world
M1	Δίνουμε ως είσοδο …		
M2			
Προσθέστε όσες γραμμές χρειάζονται, ανάλογα με τα μονοπάτια που είχατε!			