



4^η Εργασία

Διαδικαστικά

Η εργασία είναι **αυστηρά ατομική** και αποτελεί την 4^η από τις 5 εργασίες του μαθήματος. Ως 5^η 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 4^η εργασία έχει καταληκτική ημερομηνία και ώρα παράδοση Τρίτη **7 Ιανουαρίου 2025** 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 προθεσμίας¹.**

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!**

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 x = 0, y = 10, z = 5, count = 0, temp = 1;

    do {
        printf("Insert a positive integer:");
        scanf("%d", &x);
    } while (x <= 0);

    for (int i = 1; i <= x; i++) {
        if (i % 2 == 0) {
            y += i;
        } else {
            z *= i;
        }

        if (temp % 3 == 0) {
            y += temp;
        }

        if (z > 100) {
            z = z % y;
        }

        if (x % 2 == 0 && i % 2 != 0) {
            temp *= i;
        } else {
            temp += i;
        }

        count++;
    }

    if ((count == x) && (count != 0)) {
        if (y > z) {
            y -= z;
        } else if (z > 2 * y) {
            z -= 2 * y;
        } else {
            z -= y / 2;
        }
    }

    if (temp % 2 == 0) {
        y += temp / 2;
    } else {
        z += temp;
    }

    printf("Final values: y = %d, z = %d, temp = %d\n", y, z, temp);
}

return 0;
}
```

Για το παραπάνω πρόγραμμα:

- 1) Εξηγήστε με απλά λόγια τι κάνει το πρόγραμμα. Αυτό περιλαμβάνει την κατανόηση της λογικής και της ροής του προγράμματος, συμπεριλαμβανομένων των βρόχων και των συνθηκών του.
- 2) Επιλέξτε ένα σετ από 10 τυχαίους θετικούς ακέραιους. Εκτελέστε το πρόγραμμα με αυτές τις τιμές και παρέχετε έναν πίνακα που δείχνει κάθε τιμή εισόδου και το αντίστοιχο αποτέλεσμα.



Τμήμα Μηχανικών Η/Υ & Πληροφορικής

Πανεπιστήμιο Πατρών

235577 Εξασφάλιση Ποιότητας και Πρότυπα

- 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 αριθμηση, δεν θα θεωρείται ολοκληρωμένη.
- 4) Να καταγραφούν τα βασικά μονοπάτια. Για τη δική σας διευκόλυνση λύστε την άσκηση ως εξής: α) Βρείτε τις εξαρτήσεις/η συνύπαρξης E1, E2, ... Εν 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 αναζητήστε 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 τιμές ελέγχου για κάθε μονοπάτι (αν φυσικά υπάρχουν 3 τιμές).

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



Ζητούμενο 2 (μονάδες 4)

Σας έχει ανατεθεί ο έλεγχος ενός κώδικα που αφορά ένα σύστημα διαχείρισης βιβλιοθηκών. Ο παρακάτω κώδικας έχει τις παρακάτω λειτουργίες:

- Προσθήκη βιβλίου: Προσθήκη ενός βιβλίου με τίτλο, συγγραφέα και μοναδικό ISBN.
- Αναζήτηση βιβλίου: Αναζήτηση ενός βιβλίου με βάση τον τίτλο ή το ISBN.
- Δανεισμός βιβλίου: Δανεισμός ενός βιβλίου χρησιμοποιώντας το ISBN του.
- Επιστροφή βιβλίου: Επιστροφή ενός δανεισμένου βιβλίου.

Δίνεται το παρακάτω πρόγραμμα σε Python:

```
class Library:  
    def __init__(self):  
        self.books = {}  
        self.borrowed_books = set()  
  
    def add_book(self, title, author, isbn):  
        if isbn in self.books:  
            return "Book already exists"  
        if not title or not author or not isbn:  
            return "Invalid book details"  
        self.books[isbn] = {"title": title, "author": author}  
        return "Book added successfully"  
  
    def search_book(self, query):  
        results = []  
        for isbn, book in self.books.items():  
            if query.lower() in book["title"].lower() or query.lower() in isbn:  
                results.append({"isbn": isbn, "title": book["title"], "author": book["author"]})  
        return results  
  
    def borrow_book(self, isbn):  
        if isbn not in self.books:  
            return "Book not found"  
        if isbn in self.borrowed_books:  
            return "Book already borrowed"  
        self.borrowed_books.add(isbn)  
        return "Book borrowed successfully"  
  
    def return_book(self, isbn):  
        if isbn not in self.borrowed_books:  
            return "Book not borrowed"  
        self.borrowed_books.remove(isbn)  
        return "Book returned successfully"
```

Για το παραπάνω πρόγραμμα:

- Δημιουργήστε περιπτώσεις δοκιμών μαύρου κουτιού (black box testing) με βάση τις παρεχόμενες προδιαγραφές. Κάθε περίπτωση δοκιμής θα πρέπει να περιλαμβάνει:
 - Εισαγωγή: Τα δεδομένα που χρησιμοποιούνται για τη δοκιμή.
 - Αναμενόμενη έξοδος: Το αποτέλεσμα που περιμένετε από τη συνάρτηση.
 - Αιτιολόγηση: Γιατί αυτή η περίπτωση δοκιμής είναι σημαντική.
- Αναλύστε τον κώδικα, προσδιορίστε τις απαιτήσεις κάλυψης κώδικα και επισημάνετε διακλαδώσεις και ακραίες περιπτώσεις που πρέπει να ελεγχθούν. Κάθε περίπτωση δοκιμής θα πρέπει:
 - Να καλύπτει έναν μη δοκιμασμένο κλάδο ή μονοπάτι κώδικα.
 - Να συνδέεται με συγκεκριμένες συνθήκες στον κώδικα.



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

Στο τέλος, αναφέρετε πώς αλληλοσυμπληρώνονται οι δοκιμές μαύρου και λευκού κουτιού σε αυτό το σενάριο και αν σε αυτό το σενάριο ήταν αποτελεσματικότερο το black box testing ή το white box testing, κατά τη γνώμη σας;

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



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

Υπόδειγμα λύσης Ζητούμενο 1

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

3) Αρχικά γίνεται η απαρίθμηση των κόμβων.

```
#include <stdio.h>
```

```
1 int main()
{
int x = 0, y = 10, z = 5, count = 0;
```

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!



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

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

$$V(g) = e^{-n+2} = \dots$$

Περιοχές γράφου = ... (όπως φαίνονται στο σχήμα)

$$1 + \dots + \dots + = \dots$$

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.



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

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:

Μονοπάτι	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!			



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

Υπόδειγμα λύσης Ζητούμενο 2

- 1) Για την απάντηση του Black Box Testing δίνετε σύντομες και περιεχτικές περιγραφές. Κάποια ενδεικτικά use case είναι:

Test Case ID	Συνάρτηση	Είσοδοι	Αναμενόμενο αποτελέσμα	Λόγος για την περίπτωση δοκιμής
Example: TC-BB-01	add_book	("Book A", "Author A", "123456")	"Book added successfully"	'Έγκυρη περίπτωση για την προσθήκη ενός βιβλίου.

- 2) Για την απάντηση του White Box Testing καταγράψτε συγκεκριμένους κλάδους ή διαδρομές του κώδικα που πρέπει να ελεγχθούν και προσδιορίστε τυχόν ακραίες συνθήκες ή κλάδους που απαιτούν ιδιαίτερη προσοχή. Κάποιο ενδεικτικό use case είναι:

Test Case ID	Συνάρτηση	Είσοδοι	Κλάδος ή Συνθήκη	Λόγος για την περίπτωση δοκιμής
Example: TC-WB-01	add_book	("\"", "Author A", "123456")	Κλάδος όπου ο τίτλος είναι άδειος.	Εξασφαλίζει την επικύρωση του τίτλου που λείπει.

Βεβαιωθείτε ότι όλες οι περιπτώσεις δοκιμών είναι πλήρεις, συνοπτικές και ευθυγραμμισμένες με τα καθορισμένα πρότυπα πινάκων.

Χρησιμοποιήστε συνεπή αρίθμηση για τα αναγνωριστικά των περιπτώσεων δοκιμής (π.χ. TC-BB-01, TC-WB-01).