

Εξασφάλιση Ποιότητας και Πρότυπα



UNIVERSITY OF
PATRAS
ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ

Αναφορά 4^{ης} Εργασίας

Καθηγητής: Μιχάλης Ξένος

Ζητούμενο 1^ο:

1)

Αρχικά, έχουμε την αρίθμηση των κόμβων και τις εντολές οι οποίες αντιστοιχίζονται σε κάθε κόμβο.

Κόμβος 1:

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

Κόμβος 2:

```
if (year % 400 == 0)
```

Κόμβος 3:

```
printf("%d is a leap year.\n", year);
n=-1;
```

Κόμβος 4:

```
else if (year % 100 == 0)
```

Κόμβος 5:

```
printf("%d is not a leap year.\n", year);
n=1;
```

Κόμβος 6:

```
else if (year % 4 == 0)
```

Κόμβος 7:

```
printf("%d is a leap year.\n", year);
n=-1;
```

Κόμβος 8:

```
else {
    printf("%d is not a leap year.\n", year);
    n=12;
}
```

Κόμβος 9:

```
if(year % 2 == 0)
```

Κόμβος 10:

```
printf("%d is even.\n",year);
```

Κόμβος 11:

```
printf("%d is odd.\n", year);  
n=1234;
```

Κόμβος 12:

```
while (n != 0)
```

Κόμβος 13:

```
rem = n % 10;  
rev = rev * 10 + rem;  
n /= 10;
```

Κόμβος 14:

```
printf("Reversed number = %d", rev);
```

Κόμβος 15:

```
if(n<-1)
```

Κόμβος 16:

```
n = -2;
```

Κόμβος 17:

```
while(n<0);
```

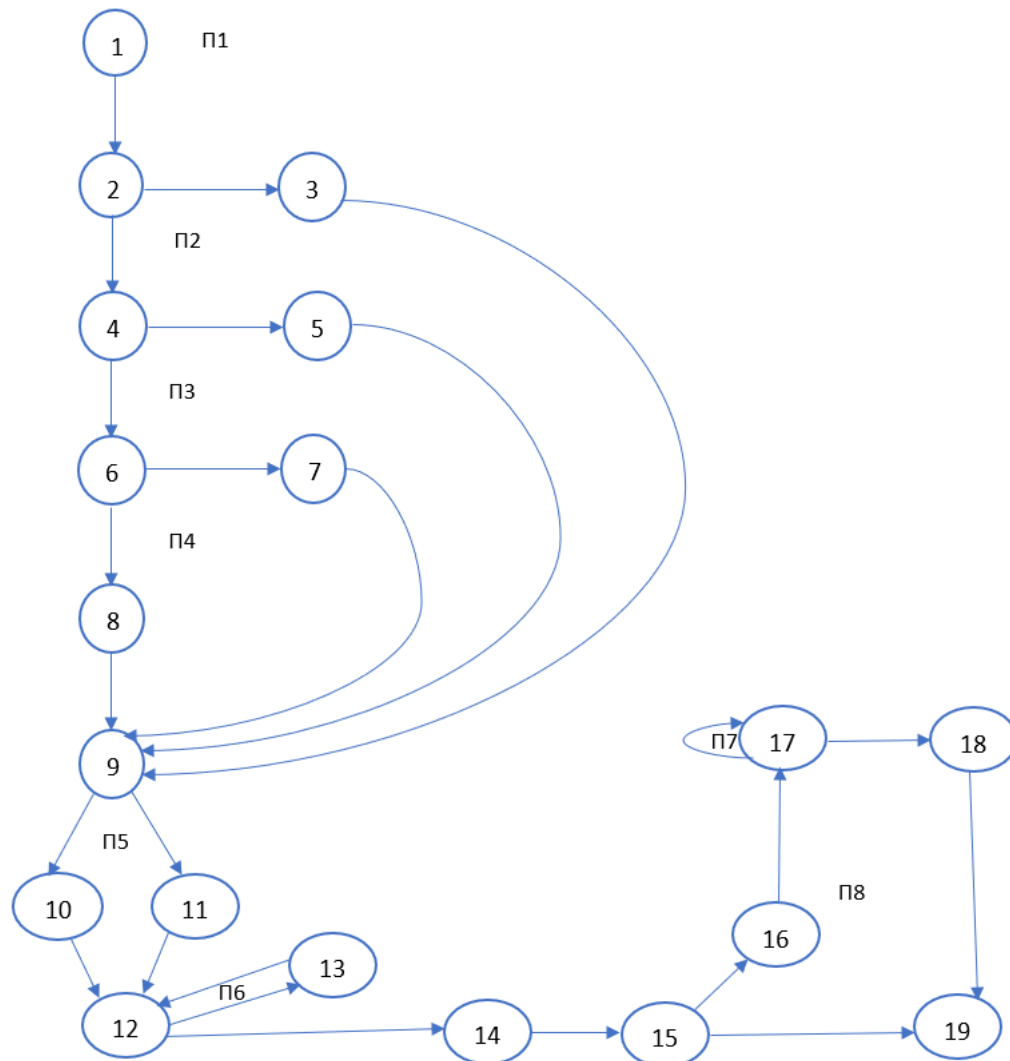
Κόμβος 18:

```
n = n + 10;
```

Κόμβος 19:

```
return 0;
```

Επομένως, με βάση την παραπάνω αρίθμηση ο γράφος ροής του προγράμματος είναι ο εξής:



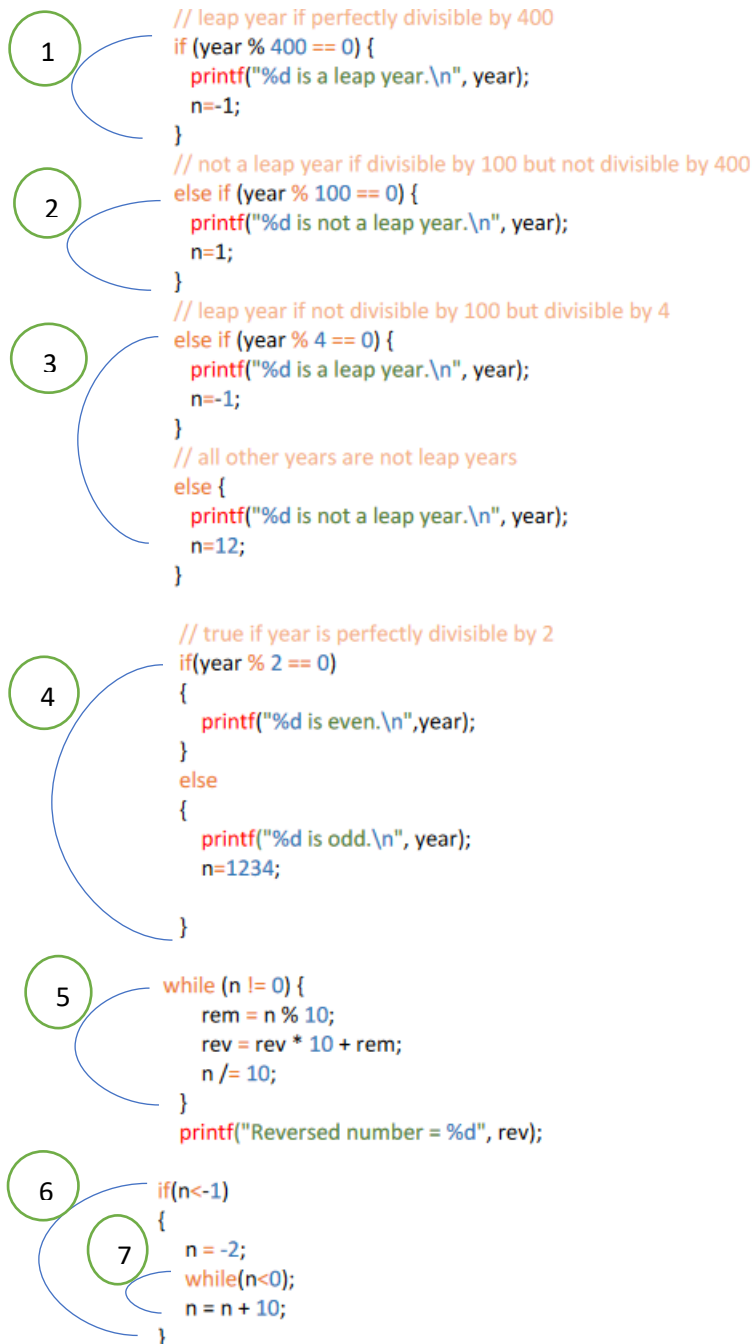
Επίσης, από τον παραπάνω γράφο προκύπτει ότι η κυκλωματική πολυπλοκότητα είναι , γιατί:

1^{ος} τρόπος : $V(g) = e - n + 2p = 25 - 19 + 2 * 1 = 8$

2^{ος} τρόπος : $V(g) = 1 + \text{εντολές} = 1 + 7 = 8$

3^{ος} τρόπος : Περιοχές γράφου = 8

Οι εντολές του προγράμματος είναι 7 οι οποίες είναι :



2)

Οι εξαρτήσεις συνύπαρξης που υπάρχουν στον γράφο του λογισμικού μας είναι :

E1: Σύμφωνα με τον κώδικα δεν θα πραγματοποιηθεί ποτέ η τελευταία “if” συνθήκη επομένως δεν θα εκτελεστούν οι κόμβοι 16 , 17 , 18 και 19.

E2: Στις περιπτώσεις που σε κάποιο μονοπάτι εκτελείται είτε ο κόμβος 3 είτε ο 5 είτε ο 7, ο κόμβος 12 θα εκτελεστεί μόνο μία φορά.

E3: Στις περιπτώσεις που σε κάποιο μονοπάτι εκτελείται είτε ο κόμβος 3 είτε ο 5 είτε ο 7, δε θα εκτελεστεί ο 11.

E4: Στις περιπτώσεις που σε κάποιο μονοπάτι εκτελούνται ο κόμβος 8 και ο 11, τότε θα εκτελεστεί και ο κόμβος 12 τέσσερις φορές.

E5: Στις περιπτώσεις που σε κάποιο μονοπάτι εκτελούνται ο κόμβος 8 και ο 10, τότε θα εκτελεστεί και ο κόμβος 12 δύο φορές.

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

M1: 1-2-3-9-10-12-13-12-14-15-19

Στη συνέχεια, ακολουθώντας τον αλγόριθμο έχουμε:

M2: 1-2-4-5-9-10-12-13-12-14-15-19

M3: 1-2-4-6-7-9-10-12-13-12-14-15-19

M4: 1-2-4-6-8-9-10-12-13-12-13-12-14-15-19

M5: 1-2-4-6-8-9-11-12-13-12-13-12-13-12-13-12-14-15-19

Σε αυτό το σημείο, οι μόνες ακμές που δεν συμπεριλαμβάνονται σε κανένα βασικό μονοπάτι είναι οι ακμές 16,17 και 18. Συνεπώς, το πρόγραμμά μπορεί να ελεγχθεί με 5 βασικά μονοπάτια, δηλαδή λιγότερα από την κυκλωματική πολυπλοκότητα η οποία αποτελεί άνω όριο των βασικών μονοπατιών.

3)

Μονοπάτι	Περιγραφή	Περίπτωση Ελέγχου (input)	Αναμενόμενο αποτέλεσμα (έξοδος προγράμματος)
M1	Δίνουμε ως είσοδο ζυγό αριθμό που διαιρείται με το 400	4000	4000 is a leap year. 4000 is even. Reversed number = -1
M2	Δίνουμε ως είσοδο ζυγό αριθμό που διαιρείται με το 100 αλλά όχι με το 400	1000	1000 is not a leap year. 1000 is even. Reversed number = 1
M3	Δίνουμε ως είσοδο ζυγό αριθμό που διαιρείται με το 4 αλλά όχι με το 100	8044	8044 is a leap year. 8044 is even. Reversed number = -1
M4	Δίνουμε ως είσοδο τυχαίο ζυγό αριθμό	5202	5202 is not a leap year. 5202 is even. Reversed number = 21
M5	Δίνουμε ως είσοδο τυχαίο μόνο αριθμό	5201	5201 is not a leap year. 5201 is odd. Reversed number = 4321

Ζητούμενο 2^ο:

1)

Αρχικά, έχουμε την αρίθμηση των κόμβων και τις εντολές οι οποίες αντιστοιχίζονται σε κάθε κόμβο.

Κόμβος 1:

```
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;
```

Κόμβος 2:

```
        try {
            Scanner s = new Scanner(new File(fileName));
```

Κόμβος 3:

```
        while (s.hasNextLine())
```

Κόμβος 4:

```
            line = s.nextLine().trim();
            number = Integer.parseInt(line);
            int temp = 0;
```

Κόμβος 5:

```
            for(int i=1;i<=number/2;i++)
```

Κόμβος 6:

```
            if(number%i == 0)
```

Κόμβος 7:

```
                temp += i;
```

Κόμβος 8:

```
            if(temp == number)
```

Κόμβος 9:

```
                System.out.println(number + " is a perfect number");
```


Κόμβος 10:

```
System.out.println(number + " is NOT a perfect number");
```

Κόμβος 11:

```
catch (FileNotFoundException ex)
```

Κόμβος 12:

```
System.out.println(fileName+": File not Found!");
```

Κόμβος 13:

```
catch (NumberFormatException ex)
```

Κόμβος 14:

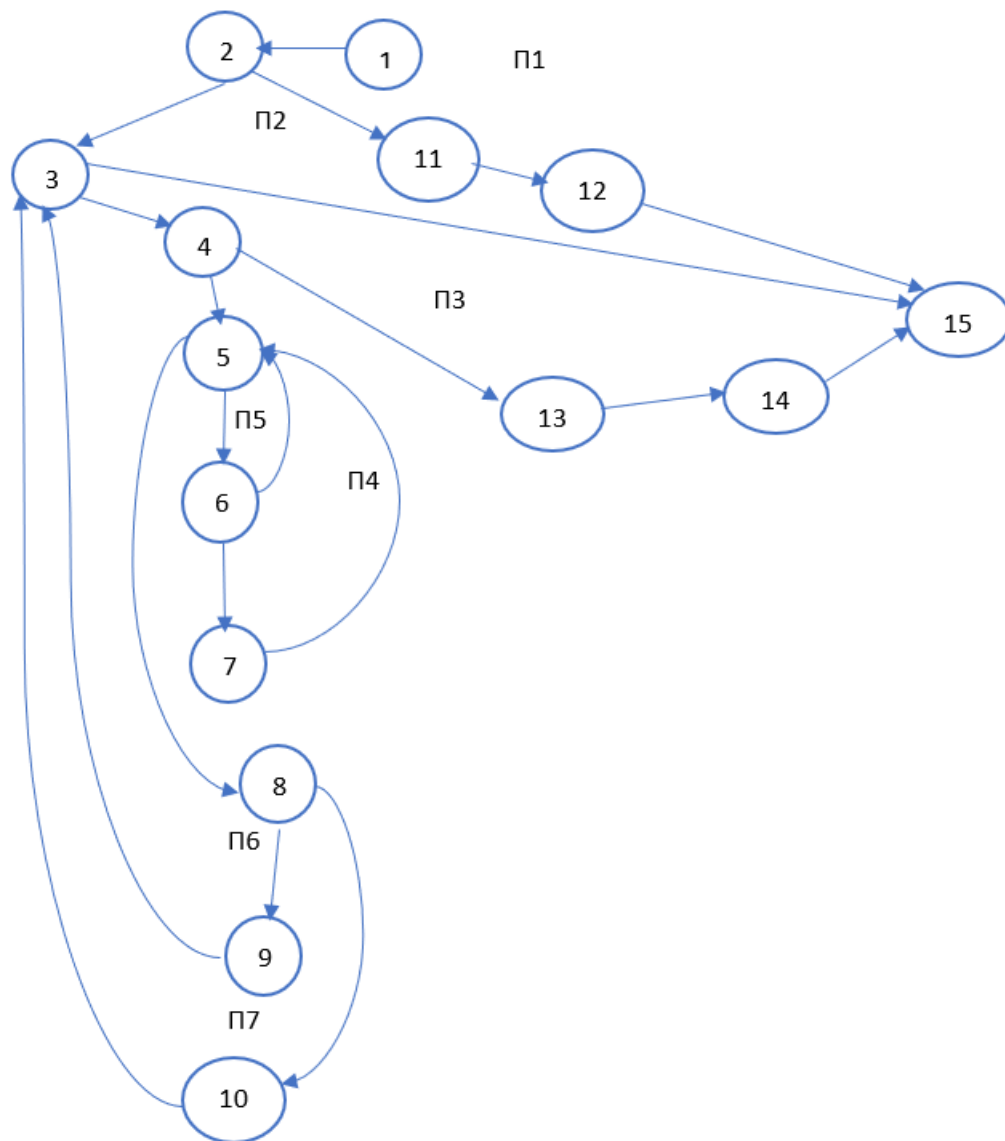
```
System.out.println(""+line+" is not a number !");
```

Κόμβος 15:

```
    }
```

2)

Επομένως, με βάση την παραπάνω αρίθμηση ο γράφος ροής του προγράμματος είναι ο εξής:



Επίσης, από τον παραπάνω γράφο προκύπτει ότι η κυκλωματική πολυπλοκότητα είναι , γιατί:

1^{ος} τρόπος : $V(g) = e - n + 2p = 20 - 15 + 2 * 1 = 7$

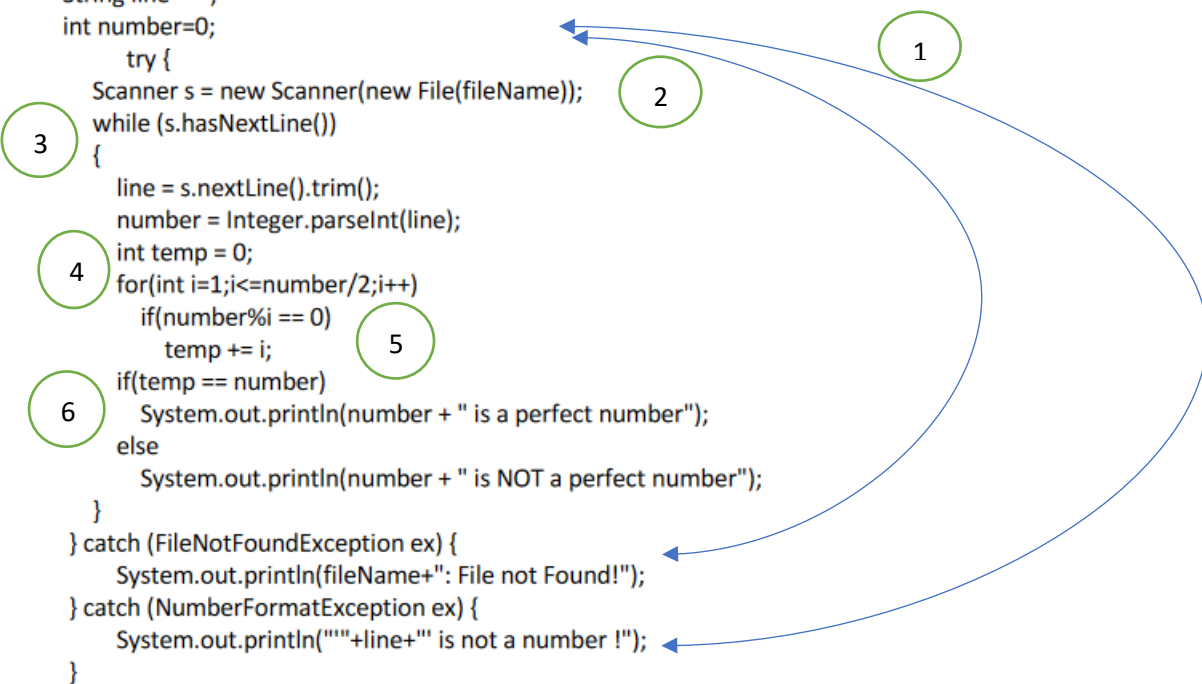
2^{ος} τρόπος : $V(g) = 1 + \text{εντολές} = 1 + 6 = 7$

3^{ος} τρόπος : Περιοχές γράφου = 7

Οι εντολές του προγράμματος είναι 7 οι οποίες είναι :

```
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 !");
        }
    }
}
```



Δηλαδή: έχουμε 2 try-catch εντολές , 2 if εντολές , 1 for εντολή και 1 while εντολή.

3)

Το μικρότερο έγκυρο μονοπάτι είναι το εξής:

M1: 1-2-11-12-15

Στη συνέχεια, ακολουθώντας τον αλγόριθμο έχουμε:

M2: 1-2-3-4-13-14-15

M3: 1-2-3-4-5-6-7-5-8-10-3-15

M4: 1-2-3-4-5-6-7-5-8-9-3-15

Συνεπώς, το πρόγραμμά μπορεί να ελεγχθεί με 4 βασικά μονοπάτια, δηλαδή λιγότερα από την κυκλωματική πολυπλοκότητα η οποία αποτελεί άνω όριο των βασικών μονοπατιών.

4)

Μονοπάτι	Περιγραφή	Περίπτωση Ελέγχου (input)	Αναμενόμενο αποτέλεσμα (έξοδος προγράμματος)
M1	Δεν υπάρχει το αρχείο	-	C:\\numbers.txt: File not Found!
M2	Το αρχείο μπορεί να διαβαστεί αλλά δεν έχει κάποιον αριθμό στη σωστή μορφή	-	“αντικείμενο” is not a number!
M3	Στο αρχείο δεν έχουμε έναν τέλειο αριθμό	3	3 is NOT a perfect number
M4	Στο αρχείο έχουμε έναν τέλειο αριθμό	4	4 is a perfect number