Εξασφάλιση Ποιότητας και Πρότυπα
4η Εργασία

Ζητούμενο 1

Το συγκεκριμένο πρόγραμμα προτρέπει τον χρήστη να εισάγει έναν ακέραιο αριθμό (μεταβλητή a) και έπειτα εκτελεί μία δομή επανάληψης (do-while) έως ότου ο χρήστης εισάγει θετικό αριθμό.
Στη συνέχεια, εκτελεί μια δομή επανάληψης (while), στην οποία φαίνεται πως αυξάνει τη μεταβλητή b, μέχρις ότου αυτή να γίνει ίση με τη μεταβλητή a. Ταυτόχρονα, μέσα στη παραπάνω δομή επανάληψης, το πρόγραμμα ελέγχει (δομή επιλογής if-else) αν η μεταβλητή c είναι μεγαλύτερη από τη μεταβλητή a. Αν ισχύει το παραπάνω, τότε το περιεχόμενο της μεταβλητής a αυξάνεται κατά ένα, διαφορετικά αυξάνεται κατά ένα το περιεχόμενο της μεταβλητής c.
Αφού τελειώσει η παραπάνω δομή επανάληψης, το πρόγραμμα ελέγχει (δομή επιλογής if-else) αν το περιεχόμενο των μεταβλητών a και b είναι ίσο και το περιεχόμενο της μεταβλητής c διαφορετικό. Αν αυτό ισχύει, τότε αυξάνεται το περιεχόμενο της μεταβλητής c κατά ένα, ειδάλλως εξισώνεται το παραπάνω περιεχόμενο με το περιεχόμενο της μεταβλητής b.
Έπειτα, το πρόγραμμα ελέγχει (δομή επιλογής if) αν το περιεχόμενο της μεταβλητής c είναι μεγαλύτερο από το πενήντα. Αν αυτό ισχύει, αυξάνεται το περιεχόμενο της παραπάνω μεταβλητής κατά a. Τέλος, εμφανίζεται στην οθόνη το περιεχόμενο της μεταβλητής c.

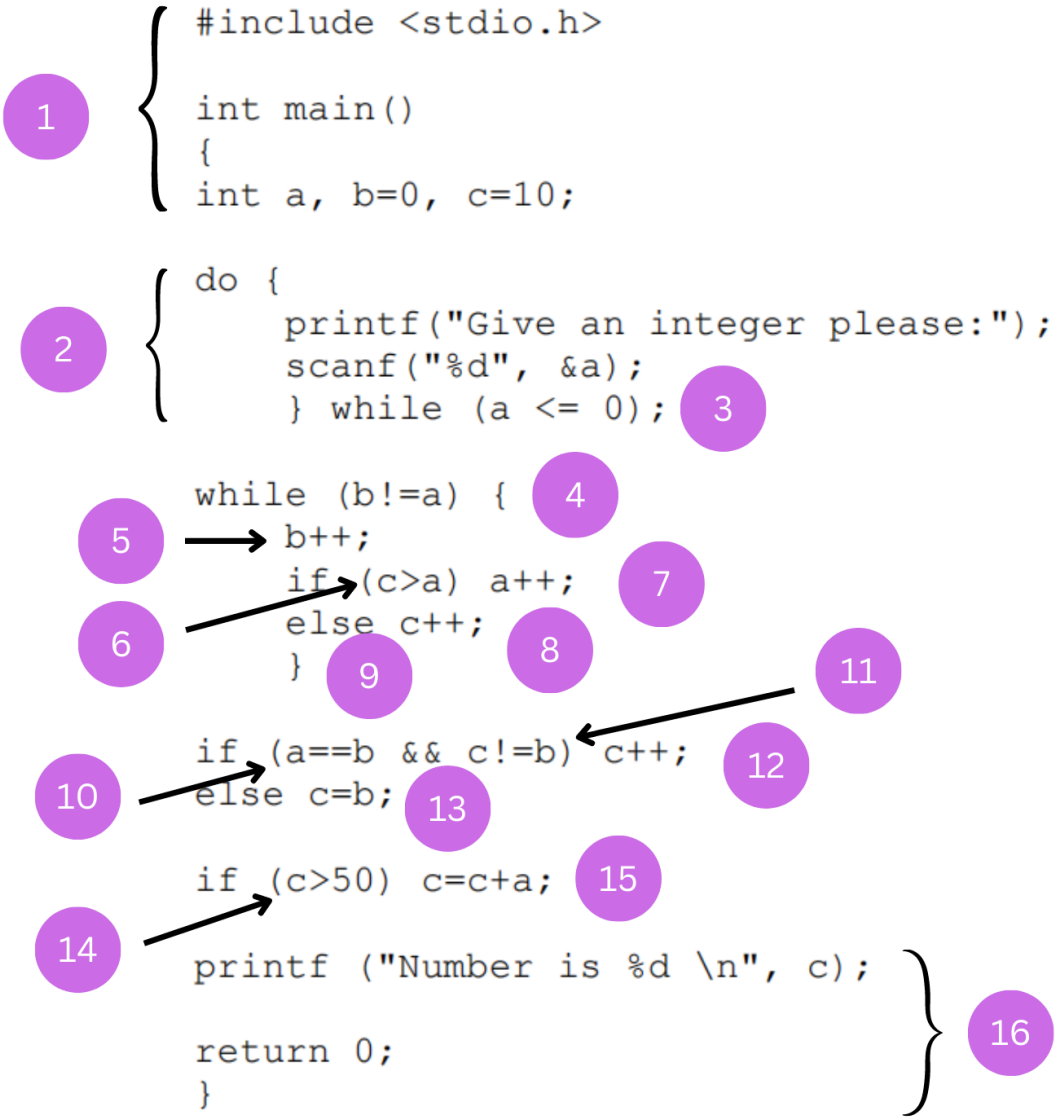
Ζητούμενο 2

Είσοδος Προγράμματος	**Έξοδος Προγράμματος**
2	Number is 13
10	Number is 21
-2 -1 5	Give an integer please: Number is 16 Give an integer please:
20	Number is 31
8	Number is 19
12	Number is 23
30	Number is 41
-10	Give an integer please:

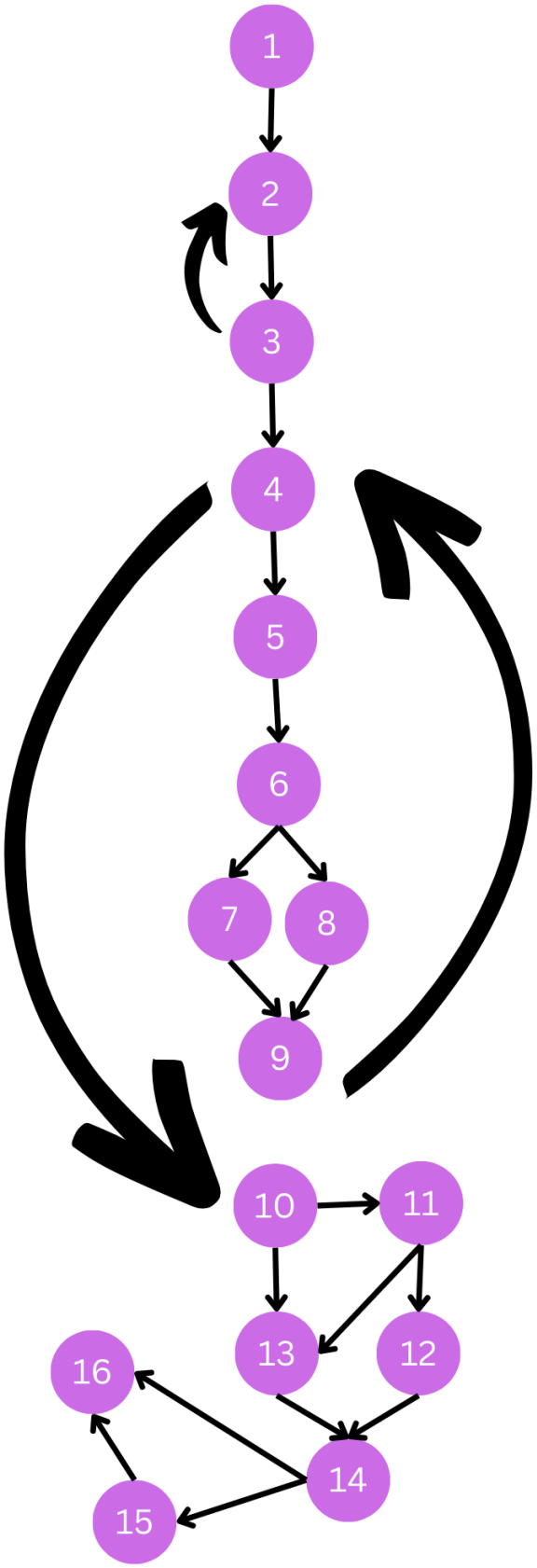
-15 -20 47	Give an integer please: Give an integer please: Number is 114
24	Number is 35
37	Number is 48

Ζητούμενο 3

Παρακάτω παρουσιάζεται ο κώδικας και οι κόμβοι, έτσι όπως έχουν αυτοί προκύψει που θα χρησιμοποιηθούν έπειτα για τη δημιουργία του ζητούμενου γράφου της κυκλωματικής πολυπλοκότητας.

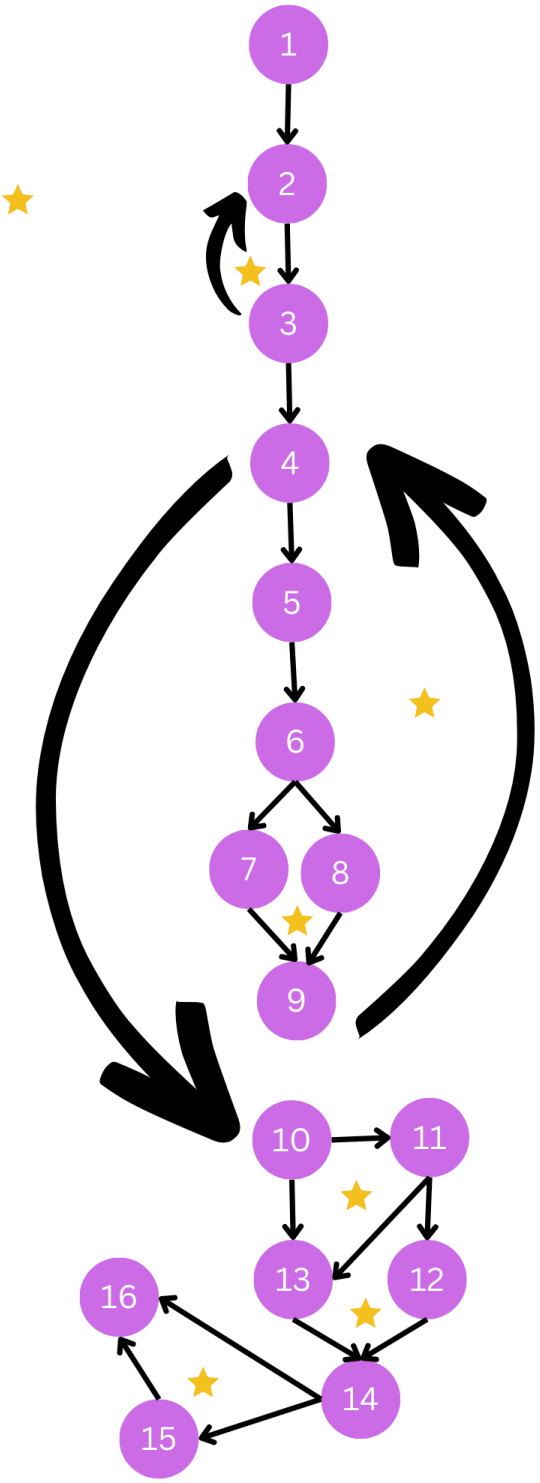


Με βάση την παραπάνω αρίθμηση ο γράφος ροής του προγράμματος είναι ο εξής:



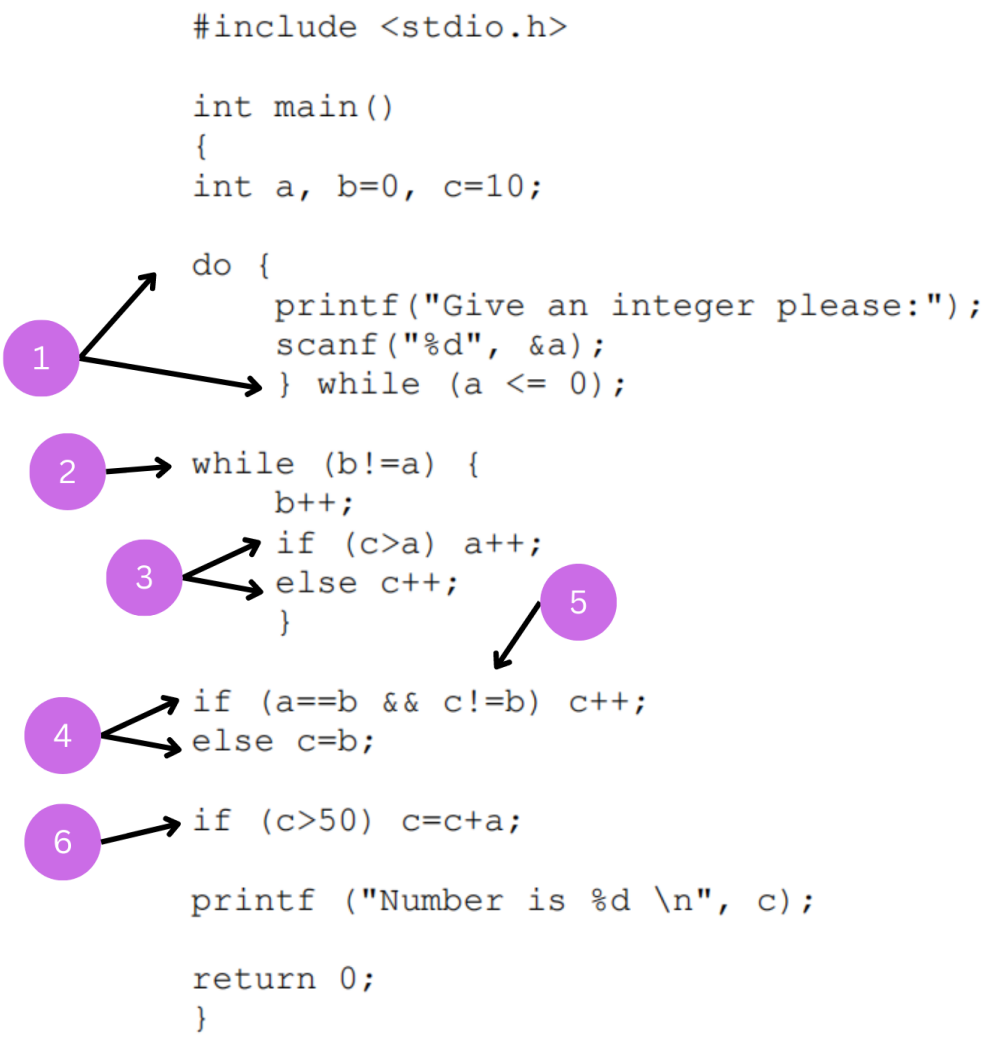
Από το γράφο προκύπτει ότι η κυκλωματική πολυπλοκότητα είναι **7**. Παρακάτω παρουσιάζονται 3 διαφορετικοί τρόποι, σύμφωνα με τους οποίους υπολογίστηκε η κυκλωματική πολυπλοκότητα.

- 1ος τρόπος:
  - V(g) = e - n + 2 * p = 21 - 16 + 2 * 1 = 7
- 2ος τρόπος:
  - V(g) = αριθμός περιοχών του γράφου = 7



Τα αστέρια στον γράφο υποδεικνύουν τις διαφορετικές περιοχές που δημιουργούνται.

- 3ος τρόπος:
  - V(g) = p + 1 = 6 + 1 = 7



Ζητούμενο 4

Για το συγκεκριμένο ερώτημα θα συνδυάσω τις τεχνικές εντοπίζοντας βασικά μονοπάτια από τον κώδικα και από το γράφο.

1. Επιλέγω το μικρότερο δυνατό μονοπάτι και ελέγχω αν είναι έγκυρο.

   M1 = 1-2-3-4-10-13-14-16

Οι κόμβοι 4-10 δεν μπορούν να συνδεθούν, αφού προκύπτει εξάρτηση (=E1). Πιο συγκεκριμένα, για να μην ισχύει η συνθήκη του while (κόμβος 4), πρέπει a=b. Ο μόνος τρόπος, για να πραγματοποιηθεί το παραπάνω, είναι b=a=0. Ωστόσο, αυτό δεν μπορεί να

ισχύει λόγω του while - κόμβος 3. Για να βγει το πρόγραμμα από αυτό το while, πρέπει οπωσδήποτε a>0.

2. Σύμφωνα με την παραπάνω εξάρτηση επιλέγω το μικρότερο δυνατό μονοπάτι που προκύπτει και ελέγχω, αν είναι έγκυρο.

    M1 = 1-2-3-4(-5-6-7-9-4)-10-13–14-16

Οι κόμβοι 10-13 δεν μπορούν να συνδεθούν, αφού προκύπτει εξάρτηση (=E2). Ειδικότερα, από τη στιγμή που σταματήσει να είναι αληθής, η συνθήκη του while - κόμβος 4, σημαίνει απαραίτητα πως a=b, επομένως η πρώτη συνθήκη του if θα ισχύει πάντοτε.

3. Σύμφωνα με τις παραπάνω εξαρτήσεις, αναζητώ το μικρότερο δυνατό μονοπάτι και ελέγχω, αν είναι έγκυρο.

    M1 = 1-2-3-4(-5-6-7-9-4)-10-11–13-14-16

    Οι κόμβοι 11 και 13 δεν μπορούν να συνδεθούν, αφού προκύπτει εξάρτηση (E3). Σύμφωνα με το while - κόμβος 4, αποκλείεται να ισχύσει b=c, συνεπώς αυτο σημαίνει πως και η συγκεκριμένη συνθήκη θα είναι συνεχώς αληθής, άρα το πρόγραμμα θα εκτελεί πάντοτε την εντολή που εμπεριέχεται στην δομή επανάληψης if.

4. Τελικά, προκύπτει το εξής βασικό μονοπάτι:

    M1 = 1-2-3-4(-5-6-7-9-4)-10-11–12-14-16

Αφού βρεθεί το πιο σύντομο βασικό μονοπάτι, αναζητώ τα επόμενα βασικά μονοπάτια σύμφωνα με την παρακάτω πρόταση. Ειδικότερα, στον πρώτο κόμβο που μπορεί να αλλάξει η διαδρομή, επιλέγω τη νέα διαδρομή και έπειτα επανέρχομαι στο σύντομο βασικό μονοπάτι (M1) στην πρώτη ευκαιρία που έχω. Κατά αυτόν τον τρόπο, προσθέτω τις λιγότερες δυνατόν ακμές. Επαναλαμβάνω την παραπάνω διαδικασία για τις επόμενες διακλαδώσεις που προκύπτουν και λαμβάνω πάντα υπόψιν μου τις εξαρτήσεις που έχουν εντοπιστεί μέχρι στιγμής αλλά και πιθανές νέες. Τελικά, προκύπτουν τα εξής βασικά μονοπάτια:

- M1 = 1-2-3-4(-5-6-7-9-4)-10-11–12-14-16
- M2 = 1-2-**3-2-3**-4(-5-6-7-9-4)-10-11–12-14-16
- M3 = 1-2-3-4(-5-**6-8-9**-4)-10-11–12-14-16
- M4 = 1-2-3-4(-5-6-7-9-4)-10-11–12-**14–15-16**

Οι μόνες ακμές που δεν συμπεριλαμβάνονται σε κανένα βασικό μονοπάτι είναι οι ακμές 10-13 και 11-13. Τελικώς, το πρόγραμμα μπορεί να ελεγχθεί με 4 βασικά μονοπάτια, δηλαδή λιγότερα από την κυκλωματική πολυπλοκότητα (=7), η οποία αποτελεί άνω όριο των βασικών μονοπατιών.

Ζητούμενο 5

Στον παρακάτω πίνακα παρατίθενται κάποιες ενδεικτικές περιπτώσεις ελέγχου για τα βασικά μονοπάτια που έχουν προκύψει από το παραπάνω ερώτημα.

Μονοπάτι	**Περιγραφή**	**Περίπτωση Ελέγχου (input)**	**Αναμενόμενο Αποτέλεσμα (έξοδος προγράμματος)**
M1	Δίνονται ως είσοδοι πρώτα το 2 και έπειτα το 5	2 5	Number is 13 Number is 16
M2	Δίνονται ως είσοδοι πρώτα το -6 και έπειτα το -10	-6 -10	Give an integer please: Give an integer please:
M3	Δίνονται ως είσοδοι πρώτα το 15 και έπειτα το 40	15 30	Number is 26 Number is 41
M4	Δίνονται ως είσοδοι πρώτα το 50 και έπειτα το 100	50 100	Number is 120 Number is 220