Task

The following program is given in C.

```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;
}
```

For the above programme:

1) Describe what the program does, i.e. give in plain language the specifications of the code. (This is a reverse engineering process).

2) Choose a set of 10 random values to confirm the correctness of your description and give a table with the value and the result (program output).

3) Construct the circuit complexity graph and based on the graph you have drawn calculate the circuit complexity in all 3 ways you have been taught. In order for the graph of circuit complexity you draw to be clear and understandable, you should indicate for each node of the graph which instructions it corresponds to. An answer that has only the graph, without numbering, will not be considered complete.

4) List the basic paths. For your convenience, solve the exercise as follows: a) Find the coexistence dependencies E1, E2, … En that exist in your graph. b) Based on these dependencies, find the shortest possible path that is valid, name it M1 and enrich it with as few new edges as possible, starting from the first node where branching is possible. If the new path is valid name it M2 and repeat (at M3, … Mn), otherwise look for another branching node. c) Continue until there are either no new edges, or no valid paths containing all edges.

5) Design control cases based on the basic path execution test technique by giving 2 control values for each path (if of course there are 2 values).

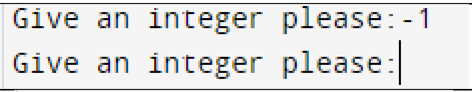
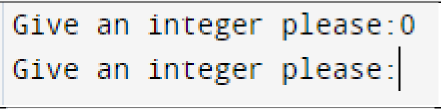
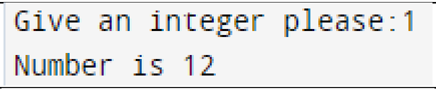
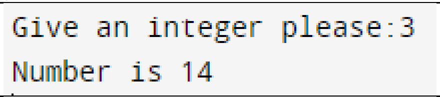
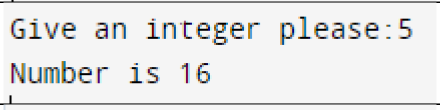
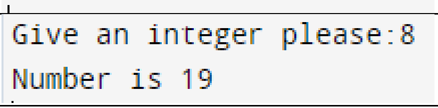
Follow the sample solution at the end of the pronunciation and present your answer in a similar way.

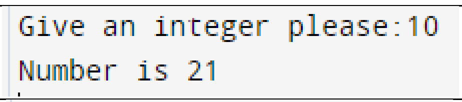
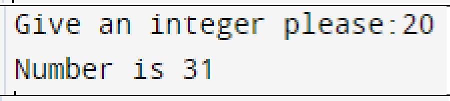
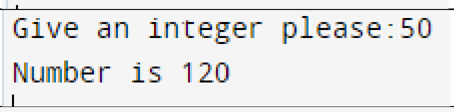
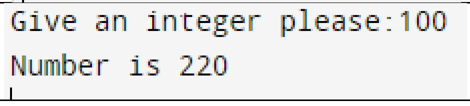
Solution

1.

First, it initializes 3 integer variables a,b,c and assigns the values 0 to b, 10 to c and leaves a empty. Then, a do-while loop follows. First, the commands inside the do will be run. A message will be printed with the printf() command, asking the user to enter an integer. The user will be able to enter this number via the scanf() command, which will ask for input from the user. The number entered by the user will be stored in the variable a. The while below contains the condition that while the number entered by the user is less than or equal to zero, the do commands will be repeated. Then follows a while statement, with the condition that while variable b is different from variable a, to increment b by 1 unit, and to check if the condition of an if statement holds. The condition says that if variable c is greater than variable a, then the value of a should increase by 1, otherwise the value of c should increase by 1. Then, an if statement follows. Its conditions are that if the value of a is equal to that of b and the value of c is different from that of b, then the value of c is to be incremented by 1, otherwise the value of b is to be entered in c. Then there is again an if command, whose condition is that if the value of c is greater than 50, then the value of a is to be added to it. Finally, there is a printf() command, which prints a message "The number is" followed by the value of the variable c.

2)

Value	**Result**
-1	 Give an integer please:-1 Give an integer please:
0	 Give an integer please:0 Give an integer please:
1	 Give an integer please:1 Number is 12
3	 Give an integer please:3 Number is 14
5	 Give an integer please:5 Number is 16
8	 Give an integer please:8 Number is 19

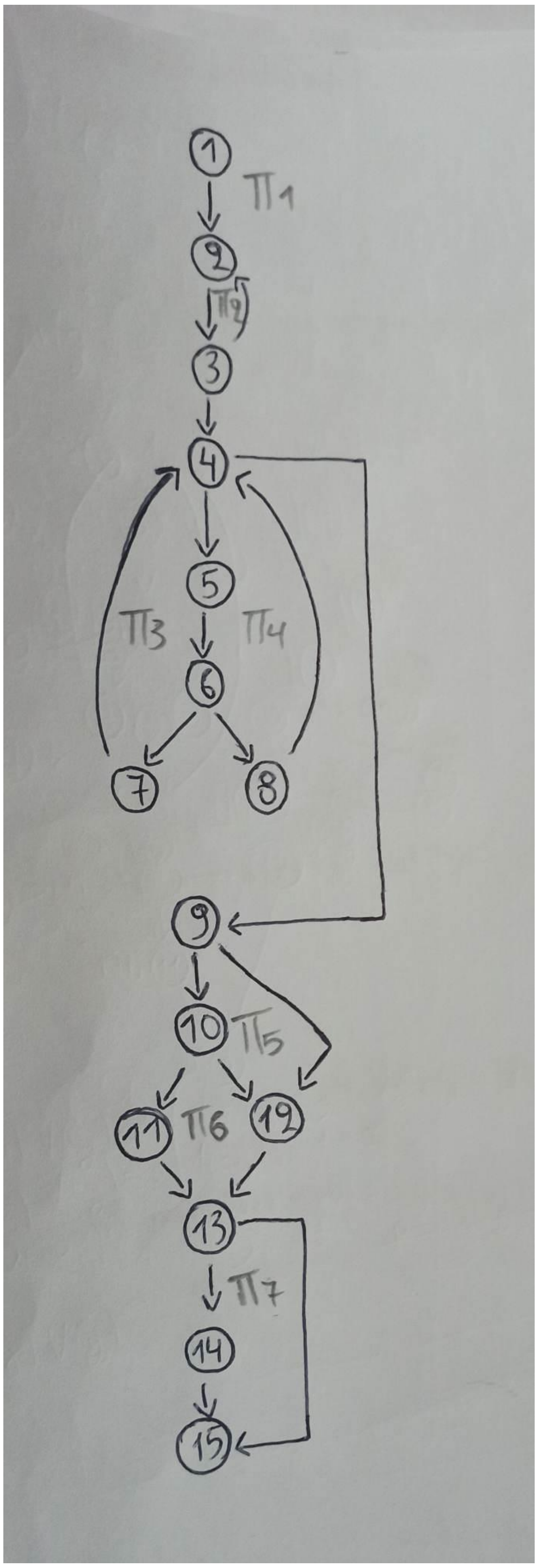
10	 Give an integer please:10 Number is 21
20	 Give an integer please:20 Number is 31
50	 Give an integer please:50 Number is 120
100	 Give an integer please:100 Number is 220

3)

Code with numbering

```
3)  #include <stdio.h>       ]
    int main()               ]
    {                        ]  1
    int a, b=0, c=10;        ]
   do {
       printf("Give an integer please:");  ]
       scanf("%d, &a);                     ]  2
 a  -> }while(a<=0); ] 3
 b ->while (b!=a){ ] 4
        b++; ] 5
     6  if( c>a) a++;  7
 c  ->  else c++; ] 8
        }
        9           10          11
 d ->if (a=5 && c!=b) c++;
                       <- e
    else c=b; ] 12
          13        14
    if (c>50) c=c+a;
 f ->
    printf("Number is %d \n", c);   ]
    return 0;                       ]  15
    }                               ]
```

Graph



Complexity calculation

(a) V(g) = e - n + 2p = 20-15+2=7

(b) V(g) = p + 1 = 6 + 1 = 7

The simple conditions p are: a, b, c, d ,e, f

(c) V(g) = graph areas = P1, P2, P3, P4, P5, P6, P7

4)

M1: 1-2-3-4-5-6-7-4-5-6-8-4-9-10-11-13-15 (shortest possible path)

M2: 1-2-3-2-3-4-5-6-7-4-5-6-8-4-9-10-11-13-15

M3: 1-2-3-4-5-6-8-4-5-6-7-4-9-10-11-13-14-15

Admission: new edges are marked in red.

At this point, the only edge not included in any basic path is 12. Formally we would have to give paths which contain this edge, but practically it would be impossible to check, so we don't need to. Therefore, the program can be checked with 3 basic paths, which is less than the circuit complexity which is an upper bound on the basic paths.

5)

M1: I give as input a=1. The result will be "Number is 12" and the path will be executed as follows:

1-2-3-(4-5-6-7)9-4-5-6-8-4-9-10-11-13-15 (fastest path in the code)

I give as input a=9. The result will be "Number is 20" and the path will be executed as follows:

1-2-3-(4-5-6-7-4-5-6-8)19-4-9-10-11-13-15 (slowest case M1)

M2: I give as input first a=0 and then a=1. The result will be for the first input "Give an integer please:" and for the second "Number is 12" and the path will be executed as follows:

1-2-3-2-3-(4-5-6-7)9-4-5-6-8-4-9-10-11-13-15

M3: I give as input a=50. The result will be "Number is 120" and the path will be executed as follows:

1-2-3-(4-5-6-8-4-5-6-7)59-4-9-10-11-13-14-15

I give as input a=100. The result will be "Number is 220" and the path will be executed as follows:

1-2-3-(4-5-6-8-4-5-6-7)109-4-9-10-11-13-14-15