LOOPS AND CONTROLS

Solutions

1. Consider the C program below.

```
#include <stdio.h>

int *A, stkTop;
int stkFunc (int opcode, int val)
{
   static int size=0, stkTop=0;
   switch (opcode)

   {
   case -1:
      size = val;
      break;

   case 0:
      if (stkTop < size ) A[stkTop++]=val;
      break;

   default:
      if (stkTop) return A[--stkTop];
   }
   return -1;
}

int main()
{
   int B[20];
   A=B;
   stkTop = -1;
   stkFunc (-1, 10);
   stkFunc (0, 5);
   stkFunc (0, 10);
   printf ("%d\n", stkFunc(1, 0)+ stkFunc(1, 0));
}
```

The value printed by the above program is _____

(a) 9 (b) 10
(c) 15 (d) 17

Solution: Option (c)

2. Which combination of the integer variables x, y and z makes the variable a get the value 4 in the following expression?

```
a = ( x > y ) ? (( x > z ) ? x : z) : (( y > z ) ? y : z )
```

(a) x = 3, y = 4, z = 2
(b) x = 6, y = 5, z = 3
(c) x = 6, y = 3, z = 5
(d) x = 5, y = 4, z = 5

Solution: Option (a)

Explanation:
The given expression assigns maximum among three elements (x, y and z) to a.

3.

```
#include<stdio.h>

int main()
{
   int a = 5;
   switch(a)
   {
   default:
      a = 4;
   case 6:
      a--;
   case 5:
      a = a+1;
   case 1:
      a = a-1;
      }
   printf("%d \n", a);
   return 0;
}
```

(a) 3
(b) 4
(c) 5
(d) None of these

Solution: Option (c)

Explanation:
There is no break statement, so first a = a + 1 is executed, then a = a-1 is executed.

4.

```
#include <stdio.h>

int main()
{
   int x = 3;
   if (x == 2); x = 0;
   if (x == 3) x++;
   else x += 2;
   printf("x = %d", x);
   return 0;
}
```

(a) x = 4
(b) x = 2
(c) Compiler Error
(d) x = 0

Solution: Option (b)

Explanation:
Value of x would be 2.

Note the semicolon after first if statement. x becomes 0 after the first if statement. So control goes to else part of second if else statement.

5.

```
#include <stdio.h>

int main()
{
   int i = 3;
   while (i--)
    {
      int i = 100;
      i--;
      printf("%d ", i);
    }
```

```
    return 0;
}
```

(a) Infinite loop
(b) 99 99 99
(c) 99 98 97
(d) 2 2 2

Solution: Option (b)

Explanation:
Note that the i–- in the statement while(i-–) changes the i of main().

And i== just after declaration statement int i=100; changes local i of while loop.

6. How many times ravindra is printed?

```
#include<stdio.h>

int main()
{
   int i = -5;
   while (i <= 5)
      {
      if (i >= 0)
         break;
      else
       {
         i++;
         continue;
      }
      printf("ravindra");
     }
   return 0;
}
```

(a) 10 times
(b) 5 times
(c) Infinite times
(d) 0 times

Solution: Option (d)

Explanation:
The loop keeps incrementing i while it is smaller than 0. When i becomes 0, the loop breaks. So ravindra is not printed at all.

7. Predict the output of the following program:

```
#include <stdio.h>

int main()
{
   int check = 20, arr[] = {10, 20, 30};
   switch (check)
   {
      case arr[0]: printf("Gates ");
      case arr[1]: printf("Quiz ");
      case arr[2]: printf("GatesQuiz");
    }
   return 0;
}
```

Solution: Option (d)

Explanation:
The case labels must be constant inside switch block. That's why the compile-time error: case label does not reduce to an integer constant is flashed.

8. What is the output of the following program?

```
#include <stdio.h>

int main()
{
   char check = 'a';
   switch (check)
         {
      case 'a' || 1: printf("Gates ");
      case 'b' || 2: printf("Quiz ");
               break;
      default: printf("GatesQuiz");
         }
   return 0;
}
```

(a) Gates
(b) Gates Quiz
(c) Gates Quiz GatesQuiz
(d) Compile-time error

Solution: Option (d)

Explanation:
An expression gets evaluated in a case label. Both the cases used are evaluated to 1(true). So compile-time error: duplicate case value is flashed as duplicated cases are not allowed.

9. In the following program, X represents the Data Type of the variable check.

```
#include <stdio.h>

int main()
{
   X check;
   switch (check)
   {
      // Some case labels
   }
   return 0;
}
```

Which of the following cannot represent X?

(a) int
(b) char
(c) enum
(d) float

Solution: Option (d)

Explanation:
A switch expression can be int, char and enum. A float variable/expression cannot be used inside switch.

10. Predict the output of the below program:

```
#include <stdio.h>

int main()
{
   int i = 3;
   switch(i)
   {
      printf("Outside ");
```

```
        case 1: printf("Gates");
           break;
        case 2: printf("Quiz");
           break;
        defau1t: printf("GatesQuiz");
       }
   return 0;
}
```

(a) Outside GatesQuiz
(b) GatesQuiz
(c) Nothing gets printed

Solution: Option (c)

Explanation:
In a switch block, the control directly flows within the case labels(or dafault label). So, statements which do not fall within these labels, Outside is not printed. Please take a closer look at the default label.

Its defau1t, not default which s interpreted by the compiler as a label used for goto statements. Hence, nothing is printed in the above program.

11. What will be the output of the following C program segment? (GATE CS 2012)

```
char inchar = 'A';

switch (inchar)
{
case 'A' :
   printf ("choice A \n") ;
case 'B' :
   printf ("choice B ") ;
case 'C' :
case 'D' :
case 'E' :
default:
   printf ("No Choice") ;
}
```

(a) No choice
(b) Choice A

(c) Choice A
Choice B No choice
(d) Program gives no output as it is erroneous

Solution: Option (c)

Explanation:
There is no break statement in case 'A'. If a case is executed and it doesn't contain break, then all the subsequent cases are executed until a break statement is found. That is why everything inside the switch is printed.

12.

```
#include <stdio.h>

int main()
{
  int i;
  for (i = 1; i != 10; i += 2)
    printf(" GatesQuiz ");
  return 0;
}
```

(a) GatesQuiz GatesQuiz GatesQuiz GatesQuiz GatesQuiz
(b) GatesQuiz GatesQuiz GatesQuiz …. infinite times
(c) GatesQuiz GatesQuiz GatesQuiz GatesQuiz
(d) GatesQuiz GatesQuiz GatesQuiz GatesQuiz GatesQuiz GatesQuiz

Solution: Option (b)

Explanation:
The loop termination condition never becomes true and the loop prints GatesQuiz infinite times.

In general, if a for or while statement uses a loop counter, then it is safer to use a relational operator (such as <) to terminate the loop than using an inequality operator (operator !=).

13. Output of following C program?

```
#include<stdio.h>

int main()
{
```

```
int i = 0;
for (printf("1st\n"); i < 2 && printf("2nd\n"); ++i && printf("3rd\n"))
   {
      printf("*\n");
   }
   return 0;

}
```

(a) 1st
2nd
*
3rd
2nd
*

(b) 1st
2nd
*
3rd
2nd
*
3rd

(c) 1st
2nd
3rd
*
2nd
3rd

(d) 1st
2nd
3rd
*
1st
2nd
3rd

Solution: Option (b)

Explanation:
It is just one by one execution of statements in for loop.

a) The initial statement is executed only once.

b) The second condition is printed before '*' is printed. The second statement also has short circuiting logical && operator which prints the second part only if 'i' is smaller than 2

c) The third statement is printed after '*' is printed. This also has short circuiting logical && operator which prints the second part only if '++i' is not zero.

14.

```
# include <stdio.h>

int main()
```

```
{
  int i = 0;
  for (i=0; i<20; i++)
   {
   switch(i)
   {
     case 0:
       i += 5;
     case 1:
       i += 2;
     case 5:
       i += 5;
     default:
       i += 4;
       break;
    }
   printf("%d  ", i);
   }
  return 0;
}
```

(a) 5 10 15 20
(b) 7 12 17 22
(c) 16 21
(d) Compiler Error

Solution: Option (c)

Explanation:
Initially i = 0. Since case 0 is true i becomes 5, and since there is no break statement till last statement of switch block, i becomes 16. Now in next iteration no case is true, so execution goes to default and i becomes 21.

In C, if one case is true switch block is executed until it finds break statement. If no break statement is present all cases are executed after the true case. If you want to know why switch is implemented like this, well this implementation is useful for situations like below.

```
switch (c)
{
   case 'a':
   case 'e':
   case 'i' :
   case 'o':
   case 'u':
```

```
    printf(" Vowel character");
    break;
  default :
    printf("Not a Vowel character");; break;
}
```

15. Output?

```
#include <stdio.h>

int main()
{
   int c = 5, no = 10;
   do {
      no /= c;
   } while(c--);
   printf ("%d\n", no);
   return 0;
}
```

(a) 1 (b) Runtime Error
(c) 0 (d) Compiler Error

Solution: Option (b)

Explanation:
There is a bug in the above program. It goes inside the do-while loop for c = 0 also. So it fails during runtime.

16.

```
#include<stdio.h>

int main()
{
  int n;
  for (n = 9; n!=0; n--)
    printf("n = %d", n--);
  return 0;
}
```

What is the output?

(a) 9 7 5 3 1
(b) 9 8 7 6 5 4 3 2 1
(c) Infinite Loop
(d) 9 7 5 3

Solution: Option (c)

Explanation:
The program goes in an infinite loop because n is never zero when loop condition (n != 0) is checked. n changes like 7 5 3 1 -1 -3 -5 -7 -9 …

17.

```
#include <stdio.h>

int i;
int main()
{
   if (i);
   else
      printf("Ëlse");
   return 0;
}
```

What is correct about the above program?

(a) if block is executed.
(b) else block is executed.
(c) It is unpredictable as i is not initialized.
(d) Error: misplaced else

Solution: Option (b)

Explanation:
Since i is defined globally, it is initialized with default value 0. The Else block is executed as the expression within if evaluates to FALSE.

Please note that the empty block is equivalent to a semi-colon(;). So the statements if (i); and if (i) {} are equivalent.

18.

```
#include <stdio.h>

int main()
{
   int i;
   if (printf("0"))
      i = 3;
   else
      i = 5;
   printf("%d", i);
   return 0;
}
```

Predict the output of above program?

(a) 3
(b) 5
(c) 03
(d) 05

Solution: Option (c)

Explanation:
The control first goes to the if statement where 0 is printed. The printf(“0″) returns the number of characters being printed i.e. 1. The block under if statement gets executed and i is initialized with 3.

19. Predict the output of the below program:

```
#include <stdio.h>
#define EVEN 0
#define ODD 1

int main()
{
   int i = 3;
   switch (i & 1)
    {
      case EVEN: printf("Even");
            break;
      case ODD: printf("Odd");
            break;
```

```
        default: printf("Default");
          }
return 0;
}
```

(a) Even
(b) Odd
(c) Default
(d) Compile-time error

Solution: Option (b)

Explanation:
The expression i & 1 returns 1 if the rightmost bit is set and returns 0 if the rightmost bit is not set. As all odd integers have their rightmost bit set, the control goes to the block labeled ODD.

20.

```
#include <stdio.h>

int main()
{
   int i = 3;
   switch (i)
   {
            case 0+1: printf("Gates");
             break;
      case 1+2: printf("Quiz");
             break;
      default: printf("GatesQuiz");
      }
   return 0;
}
```

What is the output of the above program?

(a) Gates
(b) Quiz
(c) GatesQuiz
(d) Compile-time error

Solution: Option (b)

Explanation:
Expression gets evaluated in cases. The control goes to the second case block after evaluating 1+2 = 3 and Quiz is printed.