GATE

Computer Science & Information Technology

C - Programming

Function & Storage Class

DPP: 3

Q1 Consider the following C program:

```
#include<stdio.h>
int x=12;
int f1() { int x = 25; x++; return x;}
int f2() {static int x = 50; x++; return x;}
int f3() { x *= 10; return x;}
int f4(int a, int b , int c , int d) { return a+b+c+d;}
int main()
{
   int x=1;
   x += f4(f1(), f2() , f3() , f2());
   printf("%d", x);
   return 0;
}
```

The output of the program is_____?

(A) 229 (B) 250

(C) 230 (D) 228

Q2 consider the following C-Prog.

```
#include<stdio.h>
int x=12;
int f1() { int x = 25; x++; return x;}
int f2() {static int x = 50; x++; return x;}
int f3() { x *= 10; return x;}
int f4(int a, int b , int c , int d) { static int x = 10;
x=x+10;
return x+a+b+c+d;
}
int main(){
    int x=1;
    x += f4(f1(), f2() , f3() , f2());
    x += f4(f1(), f2() , f3() , f2());
    printf("%d", x);
    return 0;
}
```

The output of the program is _____

Q3 The value of j at the end of the execution of the following C program is_____

```
#include<stdio.h>
int foo (int j){
static int x = 0;
x = x + j;
return x;
}
int main () {
  int i,j;
  for (i = 0; i <=4; i++){
    j = foo(i)+foo(i);
  }
 return 0;
}
```

Q4 Consider the following program

```
#include<stdio.h>
int foo (int j){
    static int x = 0;
    x = x + j;
    return x;
}
int main () {
    int i,j;
    for (i = 0; i <=4; i++){
        j = foo(i)-foo(i);
     }
    printf("%d", j);
    return 0;
}
```

The value printed by the program is_____

Q5 Consider the given C - Program

```
#include<stdio.h>
int foo(int y){
    static int x = 1;
    x++;
```

```
    return x+y;
}
int bar(int i){
   return ++i;
}
int main(){
    int x,y=0;
    x = foo(bar(10))+y;
    y = foo(bar(11))+x;
    printf("%d\n", (x+y));
    return 0;
}
```

The output of the program is _____ ?

Q6 consider the C - Program given below .

```
#include<stdio.h>
int foo(int y){
    static int x = 1;
    x++;
    return x+y;
}
int bar(int i){
    static int x =12;
    x++;
    return i+x;
}
int main(){
    int x,y=0;
    x = bar(foo(10))+y;
    y = bar(foo(11))+x;
    printf("%d\n", (x+y));
    return 0;
}
```

Q7

```
#include<stdio.h>
int fun(int y){
   static int x = 20;
   x++;
   return x;
 }
int main(){
   int x,y;
   x = fun(20);
   y = fun(20);
   printf("%d\n", (x+y));
   return 0;
}
```

The Output of the given program is _____

Q8

```
#include<stdio.h>
int fun(int y){
 static int x = 20;
 x++;
 return x+y;

}
int main(){
   int x,y,z;
   x = fun(20);
   y = fun(22);
   z = fun(fun(20));
   printf("%d\n", (x+y+z));
   return 0;
}
```

The Output of the above program is _____

GATE

Answer Key

Q1	(B)	Q5	41
Q2	1633	Q6	78
Q3	36	Q7	43
Q4	-4	Q8	152

Hints & Solutions

Q1 Text Solution:

program initializes a global variable x = 12 and defines four functions. f1() returns 26, f2() (static variable) returns 51 on the first call and 52 on the second, while f3() modifies the global x to 120 and returns it. The function f4() sums up its four arguments. In main(), x = 1 and the expression f4(f1(), f2(), f3(), f2()) evaluates to 26 + 51 + 120 + 52 = 249. Adding this to x, we get x = 1 + 249 = 250. Finally, printf("%d", x); prints 250, making the correct answer Option B (250).

Q2 Text Solution:

The above program initializes a global variable x = 12 and defines multiple functions, each handling variables differently. f1() returns 26 as it uses a local variable. f2(), having a static variable, returns values sequentially as 51, 52, 53, and 54 across calls. f3() modifies the global x, making it 120 in the first call and 1200 in the second. The function f4() maintains a static variable x, which starts at 10, increments by 10 on each call, and returns the sum of its arguments.In the first call to f4(f1(), f2(), f3(), f2()), the function calls resolve to f4(26, 51, 120, 52). Here, f4() updates its static x from 10 to 20 and returns 20 + 26 + 51 + 120 + 52 = 269. Since x was initialized to 1 in main(), it becomes x = 1 + 269 = 270.For the second call to f4(f1(), f2(), f3(), f2()), the function calls resolve to f4(26, 53, 1200, 54). Now, f4() updates its static x from 20 to 30 and returns 30 + 26 + 53 + 1200 + 54 = 1363. Adding this to x, we get x = 270 + 1363 = 1663.Hence ,therefore the printf("%d", x); statement prints 1663, making 1663 the correct answer.

Q3 Text Solution:

program consists of a function foo(int j), which uses a static variable x that retains its value across function calls. The function adds the input j to x and returns the updated value. Since x is static, it accumulates values across multiple calls instead of resetting each time.

In the main() function, a loop runs from i = 0 to i = 4, calling foo(i) twice in each iteration and summing their results to update j. Because foo(i) modifies x permanently, the second call within the same iteration works on the already updated value of x. This cumulative behavior affects the final value of j.

The following table illustrates the execution process:

Iteration (i)	foo(i) First Call	foo(i) Second Call	Sum (j)	Static x After Second Call
i = 0	foo(0) → x = 0 + 0 = 0	foo(0) → x = 0 + 0 = 0	0 + 0 = 0	0
i = 1	foo(1) → x = 0 + 1 = 1	foo(1) → x = 1 + 1 = 2	1 + 2 = 3	2
i = 2	foo(2) → x = 2 + 2 = 4	foo(2) → x = 4 + 2 = 6	4 + 6 = 10	6
i = 3	foo(3) → x = 6 + 3 = 9	foo(3) → x = 9 + 3 = 12	9 + 12 = 21	12
i = 4	foo(4) → x = 12 + 4 = 16	foo(4) → x = 16 + 4 = 20	16 + 20 = 36	20

At the end of the loop, the last computed value of j is 36, making it the final output of the program.

Q4 Text Solution:

The function foo(int j) contains a static variable x, which retains its value across multiple calls. Unlike normal local variables, a static variable is

initialized only once and persists throughout the execution of the program. Initially, x is set to 0, and each call to foo(j) updates it as x = x + j before returning the new value.
In the main() function, a loop iterates from i = 0 to i = 4, updating j in each iteration as foo(i) - foo(i). Since foo(i) is called twice in the same statement, the first call modifies x, and then the second call further updates x before returning its value. This results in a subtraction operation where the second call always returns a greater value than the first.The step-by-step execution of foo(i) - foo(i) shows how x keeps increasing across iterations, leading to differences of 0, -1, -2, -3, -4 for i = 0 to i = 4, respectively. The value of j is updated in each loop iteration based on these differences.
At the end of the loop, the final value stored in j is -4, which is printed by the printf statement in main().

Step-by-step execution of foo(i) - foo(i)

Iteration (i)	foo(i) first call	foo(i) Second Call	Difference (j)	Static x After Second Call
i = 0	foo(0) → x = 0 + 0 = 0	foo(0) → x = 0 + 0 = 0	0 - 0 = 0	0
i = 1	foo(1) → x = 0 + 1 = 1	foo(1) → x = 1 + 1 = 2	1 - 2 = -1	2
i = 2	foo(2) → x = 2 + 2 = 4	foo(2) → x = 4 + 2 = 6	4 - 6 = -2	6
i = 3	foo(3) → x = 6 + 3 = 9	foo(3) → x = 9 + 3 = 12	9 - 12 = -3	12
i = 4	foo(4) → x = 12 + 4 = 16	foo(4) → x = 16 + 4 = 20	16 - 20 = -4	20

Thus, the final output of the program is -4.

Q5 Text Solution:
The function foo(int y) contains a static variable x, which is initialized to 1 and retains its value across multiple calls. Each time foo() is called, x is incremented before returning x + y. The function bar(int i) simply increments i by 1 and returns it.First, bar(10) returns 11, and foo(11) updates x from 1 to 2, returning 13. Since y = 0, x = 13 + 0 = 13. Next, bar(11) returns 12, and foo(12) updates x from 2 to 3, returning 15. Now, y = 15 + 13 = 28. Finally, x + y = 13 + 28 = 41 is printed. Hence, the output of the program is 41.

Q6 Text Solution:
The function foo(10) executes first, where the static variable x starts at 1, increments to 2, and returns 12. Then, bar(12) runs, where its static x starts at 12, increments to 13, and returns 25, so x = 25.Next, foo(11) is called, where x was previously 2, increments to 3, and returns 14. Then, bar(14) runs, where x was 13, increments to 14, and returns 28, so y = 53.Now, x = 25 and y = 53. The final output is computed as x + y = 25 + 53 = 78.Thus, the program prints 78.

Q7 Text Solution:
The function fun(20) has a static variable x initialized to 20, which retains its value across function calls.
First call to fun(20): x starts at 20, increments to 21, and returns 21, so x = 21.
Second call to fun(20): x was previously 21, increments to 22, and returns 22, so y = 22. Now, x + y = 21 + 22 = 43.
Thus, the program prints 43.

Q8 Text Solution:
The function fun(y) has a static variable x initialized to 20, which retains its value across calls. In the first call fun(20), x increments to 21 and returns 21 + 20 = 41, so x = 41.In the second call fun(22), x was 21, increments to 22, and returns 22 + 22 = 44, so y = 44. Next, fun(fun(20)) executes, where the inner fun(20) runs first.Here, x was 22, increments to 23, and returns 23 + 20 = 43. Now, fun(43) executes with x = 23, which

increments to 24 and returns 24 + 43 = 67, so z = 67.Now, we compute the final sum: x + y + z = 41 + 44 + 67 = 152. Thus, the program prints 152