Week 3: Assignment Solution

1. Which of the following statement is correct?
    a) Operator precedence determines which operator is performed first in an expression with more than one operator with different precedence. Associativity is used when two operators of same precedence appear in an expression
    b) Operator associativity determines which operator is performed first in an expression with more than one operator with different associativity. Precedence is used when two operators of same precedence appear in an expression
    c) Operator precedence and associativity are same.
    d) None of the above

Solution: (a) Operator precedence determines which operator is performed first in an expression with more than one operator with different precedence, whereas associativity is used when two operators of same precedence appear in an expression

2. Find the output of the following C code

```
#include<stdio.h>
int main()
{
  int a=50, b=20, c=6, d=3, result;
  result=a+a*-b/c%d+c*d;
  printf("%d", result);
  return 0;
}
```

a) 67
b) -36
c) 66
d) -37

Solution: (a) Following the precedence rule, we can conclude that the operation steps are

- ➔ Result=50+50*- 20/6%3+6*3
- ➔ Result=50-1000/6%3+6*3
- ➔ Result=50-166%3+6*3
- ➔ Result=50-1+6*3
- ➔ Result=50-1+18
- ➔ Result=49+18
- ➔ Result=67

3. What is the output of the following C code?

```
#include <stdio.h>
int main()
{
int h = 8;
int b = 4 * 6 + 3 * 4 < h*5 ?4 : 3;
printf("%d\n", b);
return 0;
}
```

a) 0
b) 3
c) 4
d) Compilation error

Solution: (c) ‘?:’ is Conditional Expression. If Condition is true ? then value X : otherwise value Y. After simplifying the expression, we get 36<40?4:3. The condition in LHS of ? is true. Thus 4 will be stored in b.

4. Find the output of the following C code

```
#include <stdio.h>
int main()
{
int x=1;
   if ((3>5) || (2!=3))
        printf("IITKGP\n");
   else if (x&=0)
        printf("IITD\n");
   else
        printf("IITM\n");
return 0;
}
```

a) IITKGP
b) IITD and IITM
c) IITKGP and IITM
d) IITM

Solution: (a) Only the first if condition will be executed as the condition inside the if statement is true. Thus IITKGP will be printed.

5. What will be the output?

```
#include <stdio.h>
int main()
{
if ((-10 && 10) || (20 && -20))
        printf("Condition is true.");
else
        printf("Condition is false.");
 return 0;
}
```

a) Condition is true
b) Condition is false

c) Error
d) No output possible

Solution: (a) Condition is true

Any non-zero value is treated as true for condition. Consider the expressions: if((-10 && 10)||(20 && -20))
=if((1) || (1))
=if(1)

6. What is the output of the following program?

```
#include<stdio.h>
int main()
{
int i;
   if(i=0,2,3)
         printf("NPTEL ");
   else
         printf("Programming on C ");
printf("%d\n", i);
return 0;
}
```

a) Programming on C 0
b) NPTEL 0
c) NPTEL 3
d) Compilation error

Solution: (b) At first zero will assign in 'i' then comma operator returns the last value which is 3 and condition becomes true.

7. What is the output of the C program given below

```
#include <stdio.h>
int main()
{
int x = 0;
      if (x++)
printf("true\n");
   else if (x == 1)
printf("false\n");
return 0;
}
```

a) true
b) false
c) Compiler dependent
d) Compiler error

Solution: (b) ++ is a post increment operator. In x++, first 0 will be assigned and then x will be incremented by 1. Thus the next else if condition will be evaluated and false will be printed.

8. What will be the output?

```
#include<stdio.h>
int main()
{
  int x;
  x= 10==20!=30;
  printf("%d", x);
  return 0;
}
```

a) 0
b) 1
c) 10
d) 30

Solution: (b) 1

Here, operators == and != have same precedence. The associativity of both == and != is left to right, i.e, the expression on the left is executed first and moves towards the right.

Thus, the expression above is equivalent to :

((10 == 20) != 30)

i.e, (10 == 20) executes first resulting into 0 (false)

then, (0 != 30) executes resulting into 1 (true)

9. What will be the output?

```
#include <stdio.h>
int main()
{
int a = 100, b = 200, c = 300;
   if (c > b > a)
printf("TRUE");
   else
printf("FALSE");
return 0;
}
```

a) TRUE
b) FALSE
c) Syntax Error
d) Compilation Error

Solution: (b) FALSE :: (c > b > a) is treated as ((c > b) > a), associativity of '>' is left to right. Therefore the value becomes ((300 > 200) > 100) which becomes (1 > 100) thus FALSE

10. What is the output of the following C code?

```
#include <stdio.h>
int main()
{
  int y = 10;
  int z = y +(y == 10);
  printf("%d\n", z);
  return 0;
}
```

a)10
b) 11
c)20
d) Compiler error

Solution: (b) ‘==’ is a relational operator. It returns 1 if the condition is true or 0 if the condition is false. Thus y==10 will return true (which is 1) and 10+1 will be 11. Hence, the value of z will be 11.