Q.1) Max Marks: 1

What will be the output of the following code.

```
#include<stdio.h>

int main()

{

        register static int i=10;

        i=11;

        printf("%d",i);

}
```

A **Compile time error** Correct Option

Solution: (A)

Solution: Answer is Compile time error.

Static variables get memory in data area while registers are not stored in program area

ERROR: multiple storage classes in declaration specifier.

register static int i=10;//conflicting identifiers in declaration of i

i=11; // i was not declared in this scope. as previous declaration statement is giving error.

printf("%d",i); // error free but there are errors in the above lines.

So, the correct answer is compile time error.

B **some garbage value**

C **10**

D **11**

Q.2) Max Marks: 1

What will be the output of the following code.

```
#include<stdio.h>

void f(char *k)

{

        k++;

        k[2] = 'H';

        printf("%c", *k);

}

int main()

{

        char str[]="GATE";

        f(str);

}
```

A **H**

B **G**

C **A** Correct Option

Solution: (C)

Solution: Answer is A

we are just passing array of characters or string "GATE" in function f() and then we are modifying some value of this by H through its location And then simply printing content of *k but it has already come at location 1 from 0th location as we used post incrementation upon it

simply printing content of "k but it has already come at location 1 from 0th location as we used post incrementation upon it.

Here, We start execution from main() function.

int main() // starting main() function body

char str[]="GATE"; // declaring character array str and initializing it with the string "GATE".

f(str); // calling f and passing str as argument : go to the body of function f

str[0]	str[1]	str[2]	str[3]	str[4]
G	A	T	E	\0

void f(char *k) // Here character pointer *k is pointing to the location str[0]

{

K++; // post increment of character pointer

It will be increased at the end of statement. So it will start pointing to the location str[1].

k[2] = 'H'; // here, at k[2] means str[2] so replacing str[2] by 'H' so array will be as

str[0]	str[1]	str[2]	str[3]	str[4]
G	A	H	E	\0

printf("%c", *k); // As we know previous *k was pointing at str[1] so it will print content of str[1] that is 1. So here, A will be printed.

After end of this function, it will return to the main() function.

So, the correct answer is A.

D **Error**

Q.3) Max Marks: 1

What will be the output of the following c program.

```
# include <stdio.h>

    # define scanf  "%s Gate Applied Course "

          main()

          {

           printf(scanf, scanf);

          getchar();

          return 0;

          }
```

A **It will print nothing.**

B **It will print some garbage value**

C **%s Gate Applied Course**

D **%s Gate Applied Course Gate Applied Course** Correct Option

Solution: (D)

Solution: Ans is (D) %s Gate Applied Course Gate Applied Course

Explanation:
After pre-processing phase of compilation, printf statement will become.
Printf("%s Gate Applied Course", "%s Gate Applied Course")
Initially it prints same as in double quotes and then by getting %s it will print string only.So output will be like this %s Gate Applied Course Gate Applied Course

Q.4) Max Marks: 1

What will be the output of this code.

```
#include<stdio.h>

int main()

{

          int a[4]={5,6,7,8};

          int *p=a+3;

          printf("%d", p[-2]);
```

```
        return 0;
}
```

A **Error**

B **No output**

C **6** Correct Option

Solution: (C)

Solution: Ans is (C). 6

Here, we are assuming array elements are stored in array with initial address 1000 then

address	1000	1004	1008	1012
value	5	6	7	8

Here, *p = a+3 means moving pointer after crossing 3-elements

*p =a+3 =1012 = 8 // here it will point to address 1012 which is storing element 8

p[-2] = (*p)-2 = 6 // it means, from the content of this address just substracts 2.

= 8-2 =6

So answer is 6.

D **8**

Q.5) Max Marks: 1

What will be the output of the following given code.

```
#include<stdio.h>
void gate(int *, int *);
int main()
{
        int a=5;
        gate(&a,&a);
        printf("%d",a);
        return 0;
}
void gate (int *c, int *d)
{
        *c+= *d*2;
        *d*=*c-3;
}
```

A **12**

B **15**

C **180** Correct Option

Solution: (C)

here , a is initialized as a=5 let its address is 1000

Calling function gate(1000, 1000) when we go to the body of the gate ()

*c is1000 and *d is 1000

*c+ = *d*2

=> *c+ = *d *2

=> *c += 5*2(since "*" >>> "=" precedence, multiplication will be done first)

= > *c =*c+ 15 (so overall a value will be updated as we are fetching it from address)

d=*c-3

=> *d *= *c-3

=> *d* = *c -3(since "-" >>> "=" precedence, subtraction is executed first and then assignment operator)

=> *d*= 15-3 => *d* = 12 => *d = *d*12 => *d = 180

Please note that in the above evaluation if the question is being asked *d = *d**c-3, then the result would have been changed to 222 as *d**c would get executed first and then subtraction.

D compile time error

Q.6) Max Marks: 1

What will be the output of the following code.

```
#include<stdio.h>

int main()

{   int *p;

     int a[5] = { 5, 6, 7, 8, 9};

     p= &a[3];

     int *ptr = &a[2];

     int n = p - ptr;

     printf(" %d %d", n, a[n]);

     return 0 ;

}
```

A **1, 6** Correct Option

Solution: (A)

Solution: Ans is (A) 1, 6

int *p; // here p is declared as pointer variable of integer type.

int a[5] = { 5, 6, 7, 8, 9};// array is declared and initialized with the five variables.

Array: let initial address of array is 1000, considering each integer is taking 4byte.

1000	1004	1008	1012	1016
5	6	7	8	9

p= &a[3];// address of a[3] i.e. 1012 is assigned to p.

int *ptr = &a[2];// address of a[2] i.e. 1008 is assigned to ptr.

int n = p - ptr;// n = p-ptr = 1008-1004 = 4/4 = 1

printf(" %d %d", n, a[n]); // here n=1 and a[1] = 6 is printed.

So, output will be 1, 6. Every time it will give n value as 1 and prints 1, a[1].

B **2, 7**

C **2, 6**

D **1, 7**

Q.7) Max Marks: 1

What will be the output of program if you run this code on 64-bit machine.

```
#include<stdio.h>

#include<string.h>

int main()

{

     printf(" %u %u %u %u", sizeof(NULL), sizeof(""), strlen(" "),
sizeof('\0'));

     return 0;

}
```

A **8, 1, 1, 1** Correct Option

Solution: (A)

Solution: Ans is (A). 8, 1, 1, 1

Since NULL is defined as *((void*)0)*, we can think of NULL as a special pointer and its size would be equal to any pointer. If the pointer size of a platform is 4 bytes, the output of the above program would be 4. But if pointer size on a platform is 8 bytes, the output of the above program would be 8, as we are considering 64-bit machine.And, "" is 1 byte because that "empty" string has EOL character ('\0').

B 4, 1, 1, 1

C 8, 0, 0, 1

D 4, 0, 0, 1

Q.8) Max Marks: 1

What will be the output of the following given program.

```
#include <stdio.h>
main() {
    char  s1[]="Gate";
    char  s2[]="Applied";
    char  s3[]="Course";
    s1=s2; s2=s3; s3=s1;
    printf("%s",s3);
}
```

A Gate

B Applied

C Course

D Error — Correct Option

Solution: (D)

Solution: Answer is Error

Here, we are first initializing some character arrays by using strings and then assigning these with each other that is incorrect because one all three strings are of different sizes.

char s1[]="Gate"; // character array s1 is declared and initialized with the string "gate" of size 4 bytes.

char s2[]="Applied"; // character array s2 is declared and initialized with the string "Applied" of size 7 byes.

char s3[]="Course"; // character array s3 is declared and initialized with the string "Course" of size 6 bytes.

s1=s2;// here we get error as incompatible types of assignment of char[8] to char[5]

s2=s3;// here we get error as incompatible types of assignment of char[7] to char[8]

s3=s1;// here we get error as incompatible types of assignment of char[5] to char[7]

So, the correct answer is Error as this code is giving compile time error.

Q.9) Max Marks: 1

Choose the correct option for the following given code.

```
#include<stdio.h>

int x;

int main()

{

        int *y= &5;

        int *const ptr= &x;

        ptr++ ;

        (*ptr)++;

        printf("%p %d", ptr, *y);

}
```

A code will successfully be executed.

B 1- error because of uninitialized constant value

C 2-errors, because of constant values fetching and modifications. — Correct Option

Solution: (C)

Solution: answer is 2-errors, because of constant values fetching and modifications.

Here, 2 errors will be thrown as we are putting constant integer pointer under post incrementation and error because we are trying to get address of some constant that is incorrect. We can get the address of a variable which is getting some memory according to its data type.

int *y= &5; // error: lvalue required as unary '&' operand

ptr++ ; // error: increment of read-only variable 'ptr'

So, correct answer will be 2-errors, because of constant values fetching and modifications.

D 3- errors because of address increment.

Q.10) Max Marks: 1

Choose the correct option for the given following code.

```
#include<stdio.h>
int main()
{
  int a,b,c;
    a=b=10;
    b=c=50;
    ("%d %d %d", a,b,c);
    return 0;
  }
```

A It will print 10 50 50

B It will give error

C Garbage values will be printed.

D No value will be printed. Correct Option

Solution: (D)

Solution: Answer is No value will be printed.
As we haven't written statement with printf so it will not print anything but also it will not produce any error as we have written everything correctly.

int a,b,c; // here a, b, c are declared as integer type.
a=b=10; // assignment starts from right so b=10 and then a=b i.e. a is getting b value i.e. 10.
b=c=50; // assignment starts from right so c=50 and then b=c i.e. b is getting c value i.e. 50.

("%d %d %d", a,b,c); // in this statement, there is no error but we have not used printf keyword i.e. function for displaying values on output screen.So nothing will be printed.

So, the correct answer is No value will be printed.

Q.11) Max Marks: 2

What will be the output of the following program.

```
#include<stdio.h>

#include<string.h>

int main()

{

      char p[7];

      char *q = " applied";

      int length = strlen(q);

      for(int i=0; i<length ; i++)

      {

            p[i] = q [ length -i];

            printf("%s",p);

      }

      return 0;

}
```

A applied

B deilppa

C plied

D No output Correct Option

Solution: (D)

Solution: Ans is (D) No Output

Here, conversion from string constant to char

a	p	p	l	i	e	d	/0

1000	1001	1002	1003	1004	1005	1006	1007

int length = strlen(q); => so length = 8

=> p[i] = q [length -i];

=> p[0] = q[8- 0] = q[8]

/0	d	e	i	l	p	p	a

printf("%s",p); // here it will get null character in the starting so it think it is the end of string so nothing will be printed here.

Q.12) Max Marks: 2

Choose the correct option if we add all the digits and total number of characters of the resultant value(output)

```
#include <stdio.h>

int main()

{

   printf("%d",printf("%d",printf(4 + "GateAppliedCourse")));

   return 0;

}
```

A 13

B 19 Correct Option

Solution: (B)

Solution: Ans is (B)19

Here, output will be AppliedCourse132

printf(4 + "GateAppliedCourse") // print string after skipping first 4-characters.

printf("%d",printf(4 + "GateAppliedCourse")) // print total number of characters which are printed by previous printf that is 13

printf("%d",printf("%d",printf(4 + "GateAppliedCourse"))); // prints 2 as previously printed 13 which are 2-digits

So output is AppliedCourse132.

Total characters = 13

Sum of all the digits = 1+3+2 = 6

Add =19 so here 19 will be output.

C 26

D 29

Q.13) Max Marks: 2

What will be the output of the following c program.

```
#include <stdio.h>

addmult ( int ii, int jj )

{

      int kk, ll ;

      kk = ii + jj ;

      ll = ii * jj ;

      return ( kk, ll ) ;

}
```

```
main( )
{
      int i = 3, j = 4, k, l ;
      k = addmult ( i, j ) ;
      l = addmult ( i, j ) ;
      printf ( "%d %d", k, l ) ;
}
```

Choose the correct option.

A 7,7

B 12,12 — Correct Option

Solution: (B)

Solution: Ans is **(B) 12, 12**

In this question, by writing integers as ii and jj can be confusing and second we are returning two variables but we know that return can only return one value so here it will choose the right most value for returning.

K = addmult(3,4)

Ii = 3 , jj=4

Kk =3+4 = 7 and ll = 3*4 = 12

Return (7, 12) = 12 so k=12

I= addmult(3,4) and at last it will also print 12 like this only

So answer will be 12, 12

C 49,49

D Error

Q.14) Max Marks: 2

What will be the output of the given code.

```
#include<stdio.h>
int main()
{
   char str[]= "hate\nappliedcourse";
   char *ptr1, *ptr2;

   ptr1 = &str[3];
   ptr2 = str + 5;
   printf("%c", --*str -*ptr1 + *ptr2 + 2);
   printf("%s", str);
   getchar();
   return 0;
}
```

A
```
  gate
appliedcourse
```

B — Correct Option
```
  egate
appliedcourse
```

Solution: (B)

Solution: Ans is (B)
```
egate
appliedcourse
```

char str[]= "hate\nappliedcourse";
ptr1 = &str[3] = point at e
ptr2 = str + 5 = pointing at 5th location 'a'
printf("%c", --*str -*ptr1 + *ptr2 + 2);
--*str = go to content and decrement it = --h
= g = ascii of g=103
*ptr1 = content of ptr1 = e and ascii value of e = 101
*ptr2 + 2 = go to content of ptr2 = a & ascii value
= 97+2 = 99 that is ascii value of c.

--*str -*ptr1 + *ptr2 + 2 = 103- 101 + 99 = 101 ascii of e
So it will print e

printf("%s", str); // here simply str will be printed but as we have already decreased it from h to g so it will print gate and when we reached at \n then it will change line so output will be
egate
Appliedcourse

C egate

D appliedcourse

Q.15) Max Marks: 2

In the following given code, a concept of function pointer is implemented.

```
#include<stdio.h>

int a(int x)

{

     while(--x >=0)

     {

          printf(" %d", x-2);

     }

}

int main()

{

          int (*p)(int)=a;

          (*p) (6);

          return 0;

}
```

Choose the strongest right option for the above given code.

A **It gives output as 6,5,4,3,2,1**

B **It gives output as 4, 2, 0, -2, -4, -6**

C **There is no concept of function pointer in C**

D **it gives output as 3,2,1,0, -1, -2** Correct Option

Solution: (D)

Solution: Ans is (D) it gives output as 3,2,1,0, -1, -2

Function pointer concept: C allows to make such pointers that are capable of holding address of a function.

So here 6 will go to the function so we will do a(6)

--6 > =0 --5> = 0 --4 > = 0 --3 > = 0 --2 > = 0 --1 > =0 --0 > = 0

5 > =0 4>= 0 3> =0 2> = 0 1> = 0 0 > = 0 -1 > = 0 **(failed here)**

pf(5-2) pf(4-2) pf(3-2) pf(2-2) pf(1-2) pf(0-2)

3 2 1 0 -1 -2

close