1. **Which of the following are good reasons for using an Object Oriented Programming Language?.**

   a. Its Easy
   b. Its Recommended
   c. It can be thought to correct its own errors
   d. You can define your own data types
   e. None of the above

2. **The ability of a function to act in different ways for different types of parameters is called.**

   a. OOP.
   b. Encapsulation.
   c. Structures.
   d. Overloading.
   e. Option (b) and (c) are both correct.

3. **We can retrieve the address of a variable using:-**

   a. & operator
   b. * operator
   c. = operator
   d. *& operator
   e. None of the above

4. **Which one of the following is wrong syntax to use default parameters in a member function**

   a. void someFunction(char='a',int=23);
   b. void someFunction(char 'a',int 23);
   c. void someFunction(char=a,int='23');
   d. void someFunction(char a,int '23');
   e. None of the above

5. ***What will be the output of the following code?***

```
void main (void)
{
  int *pointer1, *pointer2;
  int a = 12;
  pointer1 = &a;
  pointer2 = pointer1;
  *(pointer2)++;
  *(pointer1)++;
  a++;
 cout << "(" << *pointer1 << "," << *pointer2 << "," << a << ")";
}
```

   a. (12, 12, 12)
   b. (13, 13, 13)
   c. (13, 14, 15)
   d. (15, 15, 15)
   e. (0, 0, 0)

CamScanner

6. **What will be the output of the following code at cout statement (Assume no compiler error)?**

```
void function1 (int *number);
void main (void)
{
        int variable = 100;
        int *pointer = &variable;

        function1 (&variable);
        function1 (pointer);
        cout << *pointer;
}
void function1 (int *ptr)
{
        *ptr += 30;
}
```

V [100]

*P [&V]

a. 100
b. 130
c. 160
d. 190
e. Compiler error

7. ***What will be the output of the following code?***

```
void main (void)
{
        int *pointer;
        int variable = 30;
        pointer = &variable;
        variable = 60;
        *pointer *= 3;
        cout << *pointer;
}
```

*P [&V] 60x3= 180

V [30] 60

a. Compiler error
b. 30
c. 60
d. 90
e. 180

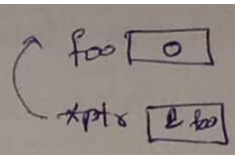
8. **The new operator**

a. Does not Returns a pointer to a variable.
b. Creates a variable called new.
c. Obtains memory for a new variable.
d. Tells how much memory is available.
e. Option (b) and (d) are both correct.

CamScanner

9. **Assume the variable declarations:**

```
int foo = 0;
int *ptr = &foo;
```



Which of the following option will change the value of foo to 1:

a. ptr++;
b. foo++;
c. (*ptr)++;
d. All of above
e. None of the above

sume that P and Q are pointers of the same type, and that each has been assigned a value. (answer 10 and 11)

10. What comparison would determine whether P and Q have targets with the same value?

a. p == q
b. &p == &q
c. *p == *q
d. All of the above
e. None of above

1. What comparison would determine whether P and Q have the same target?

a. &p == &q
b. p == q
c. *p == *q
d. All of above
e. None of the above

What is a constructor?

a. A class automatically called whenever a new object of this class is created.
b. A class automatically called whenever a new object of this class is destroyed.
c. A function automatically called whenever a new object of this class is created.
d. A function automatically called whenever a new object of this class is destroyed.
e. None of the above

Which operator is used to define a member function of a class from outside the class definition tself?

a. ::
b. :
c. >>
d. <<
e. None of the above

CamScanner

Question 2 [3 + 3 + 3 + 14 + 7 = 30 Marks]

a) Write a C++ code to define a 2-D array of size 5 x 5 by using array of pointers. [3]

b) Provide the output of the following program. [3]

```
int main(){
    int a = 7;
      int *ptr = &a;
      char ch = 'K', &cho = ch;

       cho += a;
       cho -= 5;
       *ptr += ch;
        cout << a << ", " << ch << endl;
        return 0;
}
```

c) Provide the output of the following program: [3]

```
int main(){
      int num[5];
      int* p;
      p = num;
      *p = 10;
      p+=4;
      *p = 20;
      p = &num[2];
      *p = 30;
      p = num + 1;
      *p = 40;
      p = num;
      *(p + 3) = 50;
      for (int i = 0; i < 5; i++)
            cout << num[i] << ", ";
      return 0;
      }
```


CamScanner

d) Consider the following recursive function and answer the following questions: **[14]**

```
int magic(long value)
{
        if ((value < -9) || (value > 9))
        {
                return (1 + magic(value / 10));
        }
        else
                return 1;
}
```

1) Give complete Trace and Dry Run (using functions copies and Stack) of function Magic when it is called from ***main()*** function as follows: *[8]*

```
int main()
{
    long check = 11223;
    cout<<magic(check);

    return 0;
}
```

CamScanner

2) Identify recursive case / call and base case of the magic() function? [2]

3) How many copies of magic() function will be created in the above call from main() function. [2]

4) What will be the output produced by the statement cout<<magic(check) in the main function.[1]

5) What is the purpose of magic() function? [1]

e) Write a recursive function sumofArray() that adds contents of an integer array using pointer arithmetic? **[7]**

CamScanner

Question 3 [15 Marks]

Create a class Fraction for performing arithmetic with fraction. Also, write a driver function to test your class. Use integer variables to represent the private data of the class for numerator and denominator. Provide a constructor that enables an object of this class to be initialized when it is declared. The constructor must be user defined default constructor and should store the fraction in reduced form. For example, the fraction ½ would be stored as in an object as 1 in the numerator and 2 in the denominator. Provide appropriate setter and getter functions and public member functions that perform each of the following tasks:

I. Adding two Fraction numbers. The result should be stored in Fraction object and return it.
II. Multiplying two Fraction numbers. The result should be stored in Fraction object and return it.
III. Printing Fraction numbers in the form a/b, where a is the numerator and b is the denominator.
IV. Printing Fraction numbers in floating-point format.
V. Keep the track of how many instances of the class Fraction are created.

CamScanner