Unit- I
Note : 2 Mark Questions.
Q.1 What is the most significant advantages that can we see in using references instead of pointers?
[2019]
Ans.
Reference variables are cleaner and modish as compare to the pointers; they can also be used while passing in the function as arguments, known as call by references.

Q.2 When do we to use default arguments in a function?
[2019]
Ans.
The default arguments are used when you provide no arguments or only few arguments while calling a function. The default arguments are used during compilation of program.

Q.3 Explain Structure as user defined data types.
[2018]
Ans.
A structured type is a user-defined data type that has a structure that is defined in the database. It contains a sequence of named attributes, each of which has a data type. A structured type also includes a set of method specifications. A structured type can be used as the type of a table, view, or column.

Note : 4 Mark Questions.

Q.4 Write a recursive program to find the GCD between two numbers.
[2019]
Ans.

```
#include<iostream>
using namespace std;
```

```
int gcd(int a, int b) {
   if (a == 0 || b == 0)
   return 0;
   else if (a == b)
   return a;
   else if (a > b)
   return gcd(a-b, b);
   else return gcd(a, b-a);
}
int main() {
   int a = 63, b = 42;
   cout<<"GCD of "<< a <<" and "<< b <<" is "<< gcd(a,
b);
   return 0;
}
```

Output

```
GCD of 63 and 42 is 21
```

In the above program, gcd() is a recursive function. It has two parameters i.e. a and b. If a or b is 0, the function returns 0. If a or b are equal, the function returns a. If a is greater than b, the function recursively calls itself with the values a-b and b. If b is greater than a, the function recursively calls itself with the values a and b-a.

Q.5 Elaborate the basic concepts of OOP language.
[2019]

Ans.
Object oriented programming is a type of programming which uses objects and classes its functioning. The object oriented programming is based on real world entities like inheritance, polymorphism, data hiding, etc. It aims at binding together data and function work on these data sets into a single entity to restrict their usage.

Some basic concepts of object oriented programming are −

- CLASS
- OBJECTS
- ENCAPSULATION
- POLYMORPHISM
- INHERITANCE
- ABSTRACTION

Class − A class is a data-type that has its own members i.e. data members and member functions. It is the blueprint for an object in object oriented programming language.

Object − An object is an instance of a class. It is an entity with characteristics and behaviour that are used in the object oriented programming. An object is the entity that is created to allocate memory.

Encapsulation In object oriented programming, encapsulation is the concept of wrapping together of data and information in a single unit.

Polymorphism The name defines polymorphism is multiple forms. which means polymorphism is the ability of object oriented programming to do some work using multiple forms. The behaviour of the method is dependent on the type or the situation in which the method is called.

Inheritance it is the capability of a class to inherit or derive properties or characteristics other class. it is very important and object oriented program as it allows reusability i.e. using a method defined in another class by using inheritance. The class that derives properties from other class is known as child class or subclass and the class from which the properties are inherited is base class or parent class.

C plus plus programming language supports the following types of inheritance

- single inheritance
- multiple inheritance
- multi level inheritance
- Hierarchical inheritance
- hybrid inheritance

Abstraction Data abstraction or Data Hiding is the concept of hiding data and showing only relevant data to the final user. It is also an important part object oriented programing.

Q.6 Explain the concept of defining member functions.
[2019]

Ans.

Member functions are operators and functions that are declared as members of a class. Member functions do not include operators and functions declared with the `friend` specifier. These are called *friends* of a class. You can declare a member function as `static`; this is called a *static member function*. A member function that is not declared as `static` is called a *nonstatic member function*.

The definition of a member function is within the scope of its enclosing class. The body of a member function is analyzed after the class declaration so that members of that class can be used in the member function body, even if the member function definition appears before the declaration of that member in the class member list. When the function `add()` is called in the following example, the data variables `a`, `b`, and `c` can be used in the body of `add()`.

```
class x
{
public:
      int add()             // inline member function
add
      {return a+b+c;};
private:
      int a,b,c;
};
```

Note : 12 Mark Questions.

Q.7 Explain the concept to make a private member inheritable. Also show the access mechanism in a class.
[2019]

Ans.

```
#include <iostream>
using namespace std;

class Base {
  private:
    int pvt = 1;

  protected:
    int prot = 2;

  public:
    int pub = 3;

    // function to access private member
    int getPVT() {
      return pvt;
    }
};

class PrivateDerived : private Base {
  public:
    // function to access protected member from Base
    int getProt() {
      return prot;
    }

    // function to access private member
    int getPub() {
      return pub;
    }
};

int main() {
  PrivateDerived object1;
  cout << "Private cannot be accessed." << endl;
  cout << "Protected = " << object1.getProt() << endl;
  cout << "Public = " << object1.getPub() << endl;
  return 0;
}
```

Output

```
Private cannot be accessed.
Protected = 2
Public = 3
```

Here, we have derived `PrivateDerived` from `Base` in private mode.

As a result, in `PrivateDerived`