Name: Atul Kumar Reg. No: 12102801
Roll No: RD2110B79 Question No: 01
Course code: CAP444 SET - "B"

Q1. Explain in detail the benefit of virtual function and virtual base class. Give an appropriate illustration of the program to describe both the concept in detail.

Sol": Virtual function are key to the object-oriented paradigm, such as making it easy for old code to call new code. A virtual function allows derived classes to replace the implementation provided by the base class. The compiler makes sure the replacement is always called whenever the object in question is actually of the derived class, even if the object is accessed by a base pointer rather than a derived pointer. This allows algorithms in the base class to be replaced in the derived class, even if users don't know about the derived class.

// Virtual Function & Late Binding

```
#include <iostream>
using namespace std;
class Base
{
   public:
virtual void show()
{
cout << "In Base \n"; }
};
class Derived: public Base
{
   public:
 void show()
 {
 cout << " In Derived \n"; }
 };
 int main(void)
 {
  Base * bs = new Derived;
   bs -> show();
   return 0;
}
```

Output

In Derived

→ Virtual base classes are used in virtual inheritance in a way of preventing multiple "instances" of a given class appearing in an inheritance hierarchy when using multiple inheritances. An ambiguity can arise when several paths exist to a class from the same base class. This means that a child class could have duplicate sets of members inherited from a single base class. We can solve this problem by introducing a virtual base class. When a class is made virtual, necessary care is taken so that the duplication is avoided regardless of the number of paths that exist to the child class. The virtual base class is used when a derived class has multiple copies of the base class.

```
// Virtual Base class
#include <iostream>
using namespace std;
class A {
 public:
   void f1()
    {
    cout<< "class A" <<endl;
   }
};
class B: virtual public A {
};
class C: virtual public A {
};
class D: public B, public C
{
};
 int main()
  {
    D d;
    d.f1();
   return 0;
}
```

Output

class A

Q2 Write a program to store one data of any five employees using classes in a file. Perform any six operations on the file.

Sol^n:-

```
#include <iostream>
#include <fstream>
using namespace std;
 //open(); 1st operation
 //close(); 2nd operation
 //tellp(); 3rd operation
 //seekp(); 4th operation
 //tellg(); 5th Operation
 //seekg(); 6th operation
class FileOperations
{
public:
   int empId, empSalary;
   string empName;
   void insertData()
{
   ofstream outFile;
   outFile.open("file.txt",ios::app);
   if(!outFile)
   {
```

```
cerr << "File Not Created";
    return;
}
cout<<" Enter Employee. Details Like
 empId, empName, emp salary: ";
 while(cin>>empId>>empName>>empSalary){
 outFile<<empId<<' '<<empName<<' '
          <<empSalary<<endl;
  }
  outFile.close(); //2nd operation
}
void filePointer()
{
   fstream inFile("file.txt",ios::in);
   if(!inFile)
   {
   cerr<<" File Not Available";
    return;
}
cout<<inFile.tellp()<<endl;//3rd Operation
inFile.seekp(2,ios::beg);//4th Opeation
cout<<inFile.tellp()<<endl;
cout<<inFile.tellg()<<endl;//5th operation
```

```
inFile.seekg(0, ios::beg); // 6th Operation
 string data;
 inFile >> data;
 cout << data;
  inFile.close();
}
 void printData
 {
  fstream inFile("file.txt", ios::in);
  if(!inFile)
  {
 cerr << "File Not Available";
  return;
}
 while (inFile>>empId>>empName>>
                empSalary) {
 cout<< empId << "  " << empName << ' '
          << empSalary << endl;
}
   inFile.close();
 }
};
```

```
int main()
{
  FileOperations fo;
  fo.insertData();
  fo.printData();
  fo.filePointer();
  return 0;
}
```

Output (First we have to input details)

File Name - file.txt

content in the file

101 Atul 56500
102 Vikas 44680
103 Shashank 65000
104 Reyaj 84000
105 Ayush 78650

Q3. Discuss the following concept with appropriate example:

- command line argument

Ans: The most important function of C++ is main() function. It is mostly defined with a return type of int and without parameters

```
int main() {/*... */}
```

We can also give command-line argument in C++. command line argument are given after the name of the program in command-line shell of operating systems. To pass command line arguments, we typically define main() with two arguments: first argument is the number of command line arguments and second is list of command-line arguments.

- argc() - It stores number of command line arguments.
- argv() - It is a array of character pointers listing all the arguments.

Ex- //Command Line Argument

```
#include <iostream>
using namespace std;
int main(int argc, char** argv)
{
cout<<"You have entered"<<argc
        <<"arguments:"<<"\n";
for(int i=0; i<argc; ++i)
cout<< argv[i] <<"\n";
return 0;
}
```

Output

```
You have entered 1 arguments:
./a.out
```

- This pointer: Every object in C++ has access to its own address through an important pointer called this pointer. The this pointer is an implicit parameter to all member function. Therefore, inside a member function, this may be used to refer to the invoking object.
Friend functions do not have a this pointer, because friends are not members of a class. Only member function have a this pointer. The this pointer is passed as a hidden argument to all nonstatic member function calls and is available as a local variable within the body of all nonstatic functions. This pointer is not available in static member functions as static member function can be called without any object.

Ex- // This pointer

```
#include <iostream>
using namespace std;
class Addition {
  private:
    int x;
    int y;
  public:
    void Add(int x, int y)
    {
        this->x = x;
        this->y = y;
    }
    void display() {
  cout<<"Addition:"<<x+y<<endl;
    }
};
    int main {
    Addition ad;
    ad.Add(10,30);
    ad.display();
    return 0;
}
```

Output

Addition : 40

- Pointer to derived class: A derived class is a class which takes some properties from its base class. It is true that a pointer of one class can point to other class, but classes must be a base and derived class, then it is possible. To access the variable of the base class, base class pointer will be used. So, a pointer is type of base class, and it can access all public function and variables of base class since pointer is of base class. In this pointer base class is owned by base class but points to derived class object. Same works with derived class pointer, value is changed.

Ex- // Pointer to derived class

```
#include<iostream
using namespace std;
class Base{
 public:
   int baseVar=10;
   void display(){
cout<<"I am Base class and variable
        of base class is "<<baseVar<<endl;
}};
class Derived : public Base{
  public:
   int derivedVar;
      void display(){
cout<<" I am Derived class and variable
        of derived class is "<<derivedVar<<endl;
}
};
  int main(){
    Base *bs;
    Derived dv;
    bs = &dv; // pointer base class pointing
                 to derived class
    bs->display();
    bs->baseVar=300;
    bs->display();
    return 0;
}
```

Output

I am Base class and variable of base class is 10

I am Base class and variable of base class is 300