GROUP 3

Names:

1. Nafisa Abdelaziz
2. Hossam Mohamed
3. Bassant Selima
4. Ashraf Abdulkhaliq

What is Polymorphism ?

-Polymorphism in C++ means, the same entity (function or object) behaves differently in different scenarios. Like (+)operator it performs (additions) and (concatenation).

Types of Polymorphism in C++ :

1. Static Polymorphism (Compile Time)

I. Function Overloading :

Is more than function with the same name but have different parameters.

Example :

```
#include <iostream>
using namespace std;
class Addition {
public:
     int ADD(int X,int Y)
     {
          return X+Y;
     }
     int ADD() {
          string a= "HELLO";
          string b="SAM";
         string c= a+b;
         cout<<c<<endl;

     }
};
int main()
{
     Addition obj ;
     cout<<obj.ADD(128,15)<<endl;
     obj.ADD();
     return 0;
}
```

2. Operator Overloading :

means defining additional tasks to operators without changing its actual meaning. We do this by using operator function. to provide the operators with a special meaning for a data type.

Example :

```
#include <iostream>
using namespace std;
class Complex {
private:
        int real, imag;
public:
        Complex (int r = 0,int i = 0) {
                real = r;
                imag = i; }
        Complex  operator+ ( Complex const& obj) {
                Complex res;
                res.real = real + obj.real;
                res.imag = imag + obj.imag;
                return res;}
        void print(){
        cout << real << " + i" <<imag << endl;  } };

int main()
{
        Complex c1(10, 5), c2(2, 4);
        Complex c3 = c1 + c2;
        c3.print();
        return 0; }
```

2. Runtime Polymorphism :

A vritual Function :

is a member function that is declared in the base class using the keyword virtual and is re-defined (Overriden) in the derived class.

Example :

```
#include<iostream>
using namespace std;
class Add{
public:
   virtual void print () {
      int a=20, b=30;
    cout<< " base class Action is:"<<a+b <<endl; }
  void show () {
     cout<< "show base class" <<endl; } };
 class Sub: public Add {
public:
   void print () {
     int x=20,y=10;
     cout<< " derived class Action:"<<x-y <<endl; }
   void show () {
     cout<< "show derived class" <<endl; }
 };
int main() {
   Add *aptr;
   Sub s;
   aptr = &s;
   aptr->print();
  aptr->show();
  return 0;
}
```