DATE: / /

PAGE NO.:

Assignment -3

Q.1 What is inline function? Explain with example.

Ans. C++ inline function is powerful concept that is commonly used with classes. If a function is inline, the compiler places a copy of the code of that function at each point where the function is called compile time.
To inline a function, place the keyword inline before the function name and define the function before any calls are made to the function

E.g:-
```
inline int Max(int x, int y)
{
    return (x>y) ? x : y;
}
int main()
{
    cout<<"Max(20,10): "<<Max(20,10)<<endl;
    cout<<"Max(0,200): "<<Max(0,200)<<endl;
    cout<<"Max(100,1010): "<<Max(100,1010)<<endl;

    return 0;
}
```

Q.2 Write a program to describe the concept of friend function.

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

```
class B;
class A{
 public: void showB(B&);
};
class B{
 private : int b;

public : B()
        {
          b=0;
        }
      friend void A::showB(B&x);
};
void showB(B & x)
 {
      cout << " B = " << x.b;
 }
 int main()
  {
      A a;
      B x;
      a.showB(x);
      return 0;
  }
```

Q.3 Describe all access specifier.

Ans. There are 3 types of ~~specier~~ Access Specifier:

1. Public : All the class member declared under public will be available to everyone. The data members and member functions declared public can

be accessed by other classes too. The public members of a class can be accessed from anywhere in the program using the direct member access operator (.) with the object of that class.

2. Private: The class member declared as private can be accessed only by the functions inside the class. They are not allowed to be accessed directly by any object or func^n outside the class. Only member func^n & friend func^n are allowed.

3. Protected: It is similar to that of private access modifiers, the difference is that the class member declared as Protected are inaccessible outside the class but they can be accessed by any subclass of that class.

Q.4. What is Friend function? Write a program to swap private data of two different class.

Ans. It is a function i.e not a member of our class but it has granted all the rights to access private part of the class.

```
Class digit
class number {
        int a;
    public: void read() {
                cout << " Enter a number ";
                cin >> a;
            }
            void show() {
                cout << "A = " << a << "\n";
            }
```

```
    friend void swap(digit d, number n);
};
class digit {
    int b;
    public: void read(){
                cout<<"Enter a number :";
                cin>>b;
            }
            void show(){
                cout<<"B = "<<b<<"\n";
            }
            friend void swap(digit d, number n);
};
void swap(digit d, number n){
    int temp;
    temp = n.a;
    n.a = d.b;
    d.b = temp;
    cout<<" After Swapping :";
    cout<<"A = "<<n.a<<"\t"<<"B = "<<d.b;
}
int main(){
    number n1; digit d1;
    n1.read(); d1.read();
    cout<<"\n Before swapping :\n";
    n1.show();
    d1.show();
    swap(d1, n1);
    getch();
}
```

Assignment - 4

Q.1 Write a program to find area of rectangle, circle and square using function Overloading.

Ans:

```
#include<iostream>
using namespace std;
 int area (int);
 int area (int, int);
 float area(float);

 int main (){
    int s, l, b;
    float r;
    cout<<"Enter side of a square: ";
    cin>>s;
    cout<<"Enter length and breadth of rectangle: ";
    cin>>l>>b;
    cout<<" Enter radius of circle: ";
    cin>>r;
    cout<<" Area of square "<<area(s);
    cout<<"\n Area of rectangle is "<<area(l,b);
    cout<<" Area of circle is "<<area(r);
}
 int area (int s)
 {
     return (s *s);
 }
 int area (int l, int b){
     return (l * b);
 }
 float area (float r){
```

```
    return (3.14*r*r);
}
```

Q.2 Describe over operator overloading. List the operators which cannot be overloaded. Create a class to perform matrix addition using operator overloading.

Ans: overloaded operator are functions with special names, the keyword "operator" followed by the symbol for the operator being defined. Like any othe func", an overloaded operator has a return type & a parameter list.

```
Box operator+(const Box &);
```

```
::
.*      → These operators, can't be overloaded.
.
?:
```

```
class matrix {
    int a[3][3];
    public: void accept();
            void display();
            void operator+(Matrix x);
};
void Matrix::accept(){
    cout<<"Enter Matrix element:";
    for(int i=0; i<3; i++){
        for(int j=0; j<3; j++){
            cout<<" ";
```

```
            cin >> a[i][j];
        }
    }
}
void Matrix :: display(){
    for(int i=0; i<3; i++){
        for(int j=0; j<3; j++){
            cout << a[i][j] << "\t";
        }
        cout << "\n";
    }
}
void Matrix :: operator +(matrix x)
{
    int mat[3][3];
    for(int i=0; i<3; i++){
        for(int j=0; j<3; j++){
            mat[i][j] = a[i][j] + x.a[i][j];
        }
    }
    cout << "Addition of Matrix : ";
    for(int i=0; i<3; i++)
    {
        for(int j=0; j<3; j++){
            cout << mat[i][j] << "\t";
        }
        cout << "\n";
    }
}
int main(){
```

```
Matrix m,n;
m.accept();
n.accep();
cout<<"\n First Matrix :";
m.display();
cout<<"\n Second Matrix :";
n.display();
m+n;
return 0;
}
```

Q.3 Explain scope resolution operator with the help of an example.

Ans. It is used to define a function outside a class or when we want to use a global variable but also has a local variable withe the same name.

E.g:-

```
class Student{
    int rollno;
    public: void readshow();
};
void student::readshow(){
  cout<<" Enter rollno:";
    cin>>rollno;
    cout<<rollno;
}
Void main(){
    student S1;
    S1.readshow();
    getch();
}
```

Q.4 What is static variable and static function?

Ans. static is a keyword in C++ used to give special characterstics to an element. Static element are allocated storage only once in a program lifetime in static storage area. And have a scope till the program lifetime.

Static function is a special member func^n which is used to access only static data member, any other normal data member cannot be accessed through static member function.

Assignment-5

Q.1 What is inheritance? Explain various types of inheritance giving an example of each.

Ans. It is a process by which object of one class acquire the properties of another class of object.

1. Single Inheritance: A class is allowed to inherit from only one class i.e; one sub class is inherited by one base class only.

e.g:-
```
class vechicle {
    public: vechicle()
            {
                cout << "This is a vechicle";
            }
    };
    class car : public vechicle {
};
int main() {
    car obj;
    return 0;
}
```

2. Multiple Inheritance:- In this, a class can inherit from more than one classes i.e., one sub class is inherited from more than one base classes.

e.g:-
```
class vechicle {
    public: vechicle(){
                cout << "This is a vechicle";
    }};
```

DATE : / /
PAGE NO. :

```
class fourwheeler {
    public: ~~Four~~ fourwheeler () {
                cout << " This is 4 wheeler
                                ~~Vec~~ vehicle " ;
            }
        };
class car : public Vehicle , public fourwheeler {
                        };
int main () {
        car obj ;
        return 0;
}
```

3. Multi-level Inheritance: A derived class is created from another derived class.

```
eg: class vehicle {
        public: vehicle () {
                    cout << " This is a vehicle " ;
                }
            };
        class fourwheeler : public fourwheeler {
        public: car () {
                    cout << " car has 4 wheel " ;
                }
            };
        int main () {
            car obj ;
            return 0;
        }
```

4. Hierarchical Inheritance: More than one sub-class is inherited from a single base class i.e., more than one derived class is created from a single base class.

e.g:-
```
class vehicle {
    public : vehicle()
             {
                cout<<"This is a Vehicle";
             }
    };
    class Car: public Vehicle
    {
    };
    class Bus: public Vehicle
    {
    };
    int main (){
      car obj1;
      Bus obj2;
      return 0;
    }
```

5. Hybrid Inheritance: Combining more than one type of inheritance. Like combining Hierarchical & Multiple inheritance.

e.g:
```
class vehicle {
    public: vehicle (){
              cout<<"This is a vehicle";
    }};
```

```
class Fare {
    public: Fare(){
            cout<<"Fare of vehicle";
        }
    };
    class Car : public Vehicle
    {
    };
    class Bus: public Vehicle, public Fare
    {
    };
    int main(){
        Bus obj2;
        return 0;
    }
```

Q.2 Explain multilevel Inheritance in detail with suitable c++ code.

Ans. It occurs when a class inherits from more than one base class. So the class can inherit features from multiple base classes. using ~~no~~ If a class is derive from a class and further class derived class. This is known as multilevel inheritance.

e.g:-

```
class base {
        public: int x;
                void getdata(){
                cout<<"Enter value of x=";
                cin>>x;
            }};
```

```
class derive1 : public base {
    public : int y;
             void readdata () {
                 cout << " Enter value of y= ";
                 cin >> y;
             }
};
class derive2 : public derive1 {
    private : int z;
    public:  void indata () {
                 cout << " Enter value of z= ";
                 cin >> z;
             }
             void product () {
                 cout << " Product = " << x * y * z;
             }
};
int main ()
{
    derive2 a;
    a.getdata ();
    a.readdata ();
    a.indata ();
    a.product ();
    return 0;
}
```

Q.3 Explain stream class with suitable example.

Ans. Stream classes in C++ are used to input & output operations on files & i/o devices. These

classes have specific features and to handle input and output of the program.
The iostream.h library holds all the stream classes in the C++.

Stream classes are :-

1. Istream class : For handling input stream.

e.g:-
```
int main (){
    char x;
    cin.get(x);
    cout << x;
}
```

2. Ostream class : For handling output stream.

e.g:-
```
int main (){
    char x;
    cin.get(x);
    cout.put(x);
}
```

3. iostream :- For handling both input & ~~outs~~ output stream.

e.g:-
```
int main (){
    // this func^n displays
    // ncount character from array
    cout.write ("geeksforgeeks", 5);
}
```

4. Define all modes used in file handling.

[P.T.O] =>

1. ios::in → opens the file to read.
2. ios::out → opens the file to write.
3. ios::binary → opens the file in binary mode.
4. ios::app → opens the file and appends all the output at the end.
5. ios::ate → opens the file & moves the control to the end of the file.
6. ios::trunc → Remove the data in the existing file.
7. ios::nocreate → opens the file only if it already exist.
8. ios::noreplace → opens the file only if it does not already exist.

Q.5 Write a C++ program to add two complex number by overloading binary (+) operator.

Ans.

```
class complex {
    private: float real;
             float imag;
    public: complex () : real(0), imag(0) {}
            void input () {
                cout << " Enter real & imaginary parts respectively: ";
                cin >> real;
                cin >> imag;
            }
            complex operator + (complex c2)
            {
                complex temp;
                temp.real = real + c2.real;
```

```
    temp.imag = imag + c2.imag;
    return temp;
}
void output()
{
    if(imag<0)
        cout<<"output complex number:"<<real<<
                                        imag<<"i";
        else
        cout<<" output complex number:"<<real
                                <<"+"<<imag<<"i";
    }
};
int main()
{
    complex c1, c2, result;
    cout<<" Enter first complex number ";
    c1.input();
    cout<<" Enter second complex number ";
    c2.input();
    result = c1+c2;
    result.output();
    return 0;
}
```