Assignment 4

Name: Tanish Majumdar Roll No.: 002311001077

36. Write a class `Point` which stores coordinates in (x, y) form.

Define necessary constructor, destructor and other reader/writer functions. Now overload `-` operator to calculate the distance between two points.

```cpp
#include <bits/stdc++.h>
#define ll long long
#define vll vector<long long>
#define db double
#define vi vector<int>
using namespace std;

class Point
{
    double x, y;

public:
    Point(double x, double y) : x(x), y(y) {}
    ~Point() {}

    double operator-(Point &p)
    {
        return sqrt(pow(x - p.x, 2) + pow(y - p.y, 2));
    }
};

int main()
{
    Point p1(1, 1), p2(2, 2);
    cout << "Distance between p1 and p2: " << p1 - p2 << endl;

    return 0;
}
```

37. Design a class `Complex` that includes all the necessary functions and operators like `=, +, -, *, /`.

```cpp
#include <bits/stdc++.h>
#define ll long long
#define vll vector<long long>
#define db double
#define vi vector<int>
using namespace std;

class Complex
{
    double real, imag;

public:
    Complex(double real, double imag) : real(real), imag(imag) {}
    ~Complex() {}
    Complex operator-(Complex &c)
    {
        return Complex(real - c.real, imag - c.imag);
    }
    Complex operator+(Complex &c)
    {
        return Complex(real + c.real, imag + c.imag);
    }
    Complex operator*(Complex &c)
    {
        return Complex(real * c.real - imag * c.imag, real * c.imag + imag
* c.real);
    }
    Complex operator/(Complex &c)
    {
        return Complex((real * c.real + imag * c.imag) / (c.real * c.real
+ c.imag * c.imag), (imag * c.real - real * c.imag) / (c.real * c.real +
c.imag * c.imag));
    }
    void display()
    {
        cout << real << " + " << imag << "i" << endl;
    }
};

int main()
{
    Complex c1(1, 1), c2(2, 2);
    cout << "c1 - c2: ";
    (c1 - c2).display();
    cout << "c1 + c2: ";
    (c1 + c2).display();
    cout << "c1 * c2: ";
    (c1 * c2).display();
    cout << "c1 / c2: ";
```

```
    (c1 / c2).display();

    return 0;
}
```

38. Implement a class `Quadratic` that represents second-degree polynomial i.e. polynomial of type $ax^2 + bx + c$. The class will require three data members corresponding to a, b and c.

Implement the following:

- **a.** A constructor (including a default constructor which creates a null polynomial)
- **b.** Overload the addition operator to add two polynomials of degree 2.
- **c.** Overload `<<` and `>>` operators to print and read polynomials.
- **d.** A function to compute the value of the polynomial for a given (x).
- **e.** A function to compute the roots of the equation $ax^2 + bx + c = 0$. Remember, the root may be a complex number. You may implement a “Complex” class to represent the root of the quadratic equation.

```
#include <bits/stdc++.h>
#define ll long long
#define vll vector<long long>
#define db double
#define vi vector<int>
using namespace std;

class Complex
{
    double real, imag;

public:
    Complex(double real, double imag) : real(real), imag(imag) {}
    ~Complex() {}
    void display()
    {
        cout << real << " + " << imag << "i" << endl;
    }
};

class Quadratic
{
    int a, b, c;

public:
    Quadratic(int a, int b, int c) : a(a), b(b), c(c) {}
```

```
    ~Quadratic() {}
    Quadratic() : a(0), b(0), c(0) {}
    Quadratic operator+(Quadratic &q)
    {
        return Quadratic(a + q.a, b + q.b, c + q.c);
    }
    friend ostream &operator<<(ostream &out, const Quadratic &q)
    {
        out << q.a << "x^2 + " << q.b << "x + " << q.c << endl;
        return out;
    }
    friend istream &operator>>(istream &in, Quadratic &q)
    {
        in >> q.a >> q.b >> q.c;
        return in;
    }
    long long eval(int x)
    {
        return a * x * x + b * x + c;
    }
    void roots()
    {
        int d = b * b - 4 * a * c;
        if (d < 0)
        {
            Complex c1(-b / (2 * a), sqrt(-d) / (2 * a)), c2(-b / (2 * a),
-sqrt(-d) / (2 * a));
            cout << "Roots: ";
            c1.display();
            c2.display();
            return;
        }
        cout << "Roots: " << (-b + sqrt(d)) / (2 * a) << ", " << (-b -
sqrt(d)) / (2 * a) << endl;
    }
};

int main()
{
    Quadratic q1(1, 1, 1), q2(2, 2, 2), q3;

    return 0;
}
```

39. A program is given as follows:

```
class INT {
    int i;
```

```
public:
    INT(int a) : i(a) {}
    ~INT() {}
};

int main() {
    int x = 3;
    INT y = x;
    y++ = ++y;
    x = y;
    return 0;
}
```

Write extra functions/operators required in the `INT` class to make main program work. Provide suitable implementation for the added functions/operators.

```
#include <bits/stdc++.h>
#define ll long long
#define vll vector<long long>
#define db double
#define vi vector<int>
using namespace std;

class INT
{
    int i;

public:
    INT(int a) : i(a) {}
    ~INT() {}
    INT &operator++()
    {
        i++;
        return *this;
    }
    INT operator++(int)
    {
        INT temp = *this;
        i++;
        return temp;
    }
    INT operator=(INT a)
    {
        i = a.i;
        return *this;
    }
    operator int()
```

```
    {
        return i;
    }
    void display()
    {
        cout << i << endl;
    }
};

int main()
{
    int x = 3;
    INT y = x;
    y++ = ++y;
    x = y;
    cout << x << endl;
    y.display();

    return 0;
}
```

40. Design and implement class(es) to support the following main program.

```
int main() {
        IntArray i(10);
        for(int k = 0; k < 10; k++)
                i[k] = k;
        cout << i;
        return 0;
}
```

```
#include <bits/stdc++.h>
#define ll long long
#define vll vector<long long>
#define db double
#define vi vector<int>
using namespace std;

class IntArray
{
    int *arr;
    int size;

public:
    IntArray(int s)
    {
```

```
        size = s;
        arr = new int[size];
    }
    int &operator[](int i)
    {
        return arr[i];
    }
    friend ostream &operator<<(ostream &out, IntArray &i)
    {
        for (int k = 0; k < i.size; k++)
            out << i.arr[k] << " ";
        out << endl;
        return out;
    }
};

int main()
{
    IntArray i(10);
    for (int k = 0; k < 10; k++)
        i[k] = k;
    cout << i;
    return 0;
}
```

41. You are given a main program:

```
int main() {
        Integer a = 4, b = a, c;
        c = a+b++;
        int i = a;
        cout << a << b << c;
        return 0;
}
```

Design and implement class(es) to support the main program.

```
#include <bits/stdc++.h>
#define ll long long
#define vll vector<long long>
#define db double
#define vi vector<int>
using namespace std;

class Integer
{
    int i;
```

```
public:
    Integer(int i = 0) : i(i) {}
    Integer(const Integer &i) : i(i.i) {}
    Integer operator++(int)
    {
        Integer temp = *this;
        i++;
        return temp;
    }
    Integer operator+(Integer &i)
    {
        return this->i + i.i;
    }
    operator int()
    {
        return i;
    }
    friend ostream &operator<<(ostream &out, Integer &i)
    {
        out << i.i;
        return out;
    }
};

int main()
{
    Integer a = 4, b = a, c;
    c = a + b++;
    int i = a;
    cout << a << b << c;
    return 0;
}
```

42. Design and implement class(es) to support the following code segment.

```
Table t(4, 5), t1(4, 5);
cin >> t;
t[0][0] = 5;
int x = t[2][3];
t1 = t;
cout << t << “\n” << t1;
```

```
#include <bits/stdc++.h>
#define ll long long
#define vll vector<long long>
```

```cpp
#define db double
#define vi vector<int>
using namespace std;

class Table
{
    vector<vector<int>> v;
    int rows, cols;

public:
    Table(int r, int c) : rows(r), cols(c), v(r, vector<int>(c, 0)) {}
    vector<int> &operator[](int i)
    {
        return v[i];
    }
    friend istream &operator>>(istream &in, Table &t)
    {
        for (int i = 0; i < t.rows; i++)
        {
            for (int j = 0; j < t.cols; j++)
            {
                in >> t.v[i][j];
            }
        }
        return in;
    }
    friend ostream &operator<<(ostream &out, Table &t)
    {
        for (int i = 0; i < t.rows; i++)
        {
            for (int j = 0; j < t.cols; j++)
            {
                out << t.v[i][j] << " ";
            }
            out << "\n";
        }
        return out;
    }
};

int main()
{
    Table t(4, 5), t1(4, 5);
    cin >> t;
    t[0][0] = 5;
    int x = t[2][3];
    t1 = t;
    cout << t << "\n"
         << t1;
```

```
    return 0;
}
```

43. Design and implement class(es) to support the following code segment.

```
Index in(4), out(10);
int x = in;
int y = in + out;
in = 2;
Integer i;
i = in;
```

```
#include <bits/stdc++.h>
#define ll long long
#define vll vector<long long>
#define db double
#define vi vector<int>
using namespace std;

class Index
{
    int i;

public:
    Index(int x) : i(x) {}
    operator int() { return i; }
    Index operator+(Index &x)
    {
        return Index(i + x.i);
    }
    Index operator=(int x)
    {
        i = x;
        return *this;
    }
    operator Integer(){};
};

class Integer
{
    int i;

public:
    Integer(int x = 0) : i(x) {}
    operator int() { return i; }
    Integer(Integer &x)
```

```
    {
        i = x.i;
    }
};

Index::operator Integer()
{
    return Integer(i);
}

int main()
{
    Index in(4), out(10);
    int x = in;
    int y = in + out;
    in = 2;
    Integer i;
    i = in;
    return 0;
}
```