Object Oriented Methodologies Lab ETCT 257

SUBMITTED TO:

MS. ARUSHI JAIN

ASSISTANT PROFESSOR

SUBMITTED BY:

KHUSHI JHA

ITE2

Maharaja Agrasen Institute of Technology, PSP
Area, Sector – 22, Rohini, New Delhi – 110085

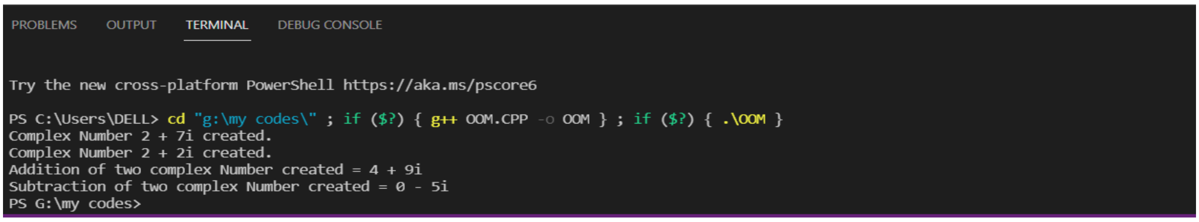
		• Function toupper() to convert lowercase letter to uppercase			
10		Create a class called LIST with functions store() and retrieve(). To store a value, call store function and to retrieve a value, call retrieve function. Derive two classes stack and queue from it and override store and retrieve.	35		
11	5	Write a program for multiplication of two matrices using OOP.	43	27/10/21	
12		Write a program to perform addition of two complex numbers using OOP.	47		

LAB1

1. **Write a program to add and subtract two complex numbers.**

```

#include <bits/stdc++.h>
using namespace std;

struct complexNumber
{
    int real;
    int imag;
    };

    void create(struct complexNumber *C, int a, int b)
    {
        C->real = a;
        C->imag = b;
        cout << "Complex Number " << C->real << " + " << C->imag << "i created." << endl;
    }

    void add(struct complexNumber *C1, struct complexNumber *C2, struct complexNumber *res)
    {
        res->real = C1->real + C2->real;
        res->imag = C1->imag + C2->imag;
        cout << "Addition of two complex Number created = " << res->real << " + " << res->imag << "i" << endl;
    }

    void subtract(struct complexNumber *C1, struct complexNumber *C2, struct complexNumber *res)
    {
        res->real = C1->real - C2->real;
        res->imag = C1->imag - C2->imag;
        cout << "Subtraction of two complex Number created = " << res->real << " - " << res->imag << "i" << endl;
    }

    int main()
    {
        struct complexNumber a, b, res;
        create(&a, 2, 7);
        create(&b, 2, 2);
        add(&a, &b, &res);
        subtract(&a, &b, &res);
        return 0;
    }

```

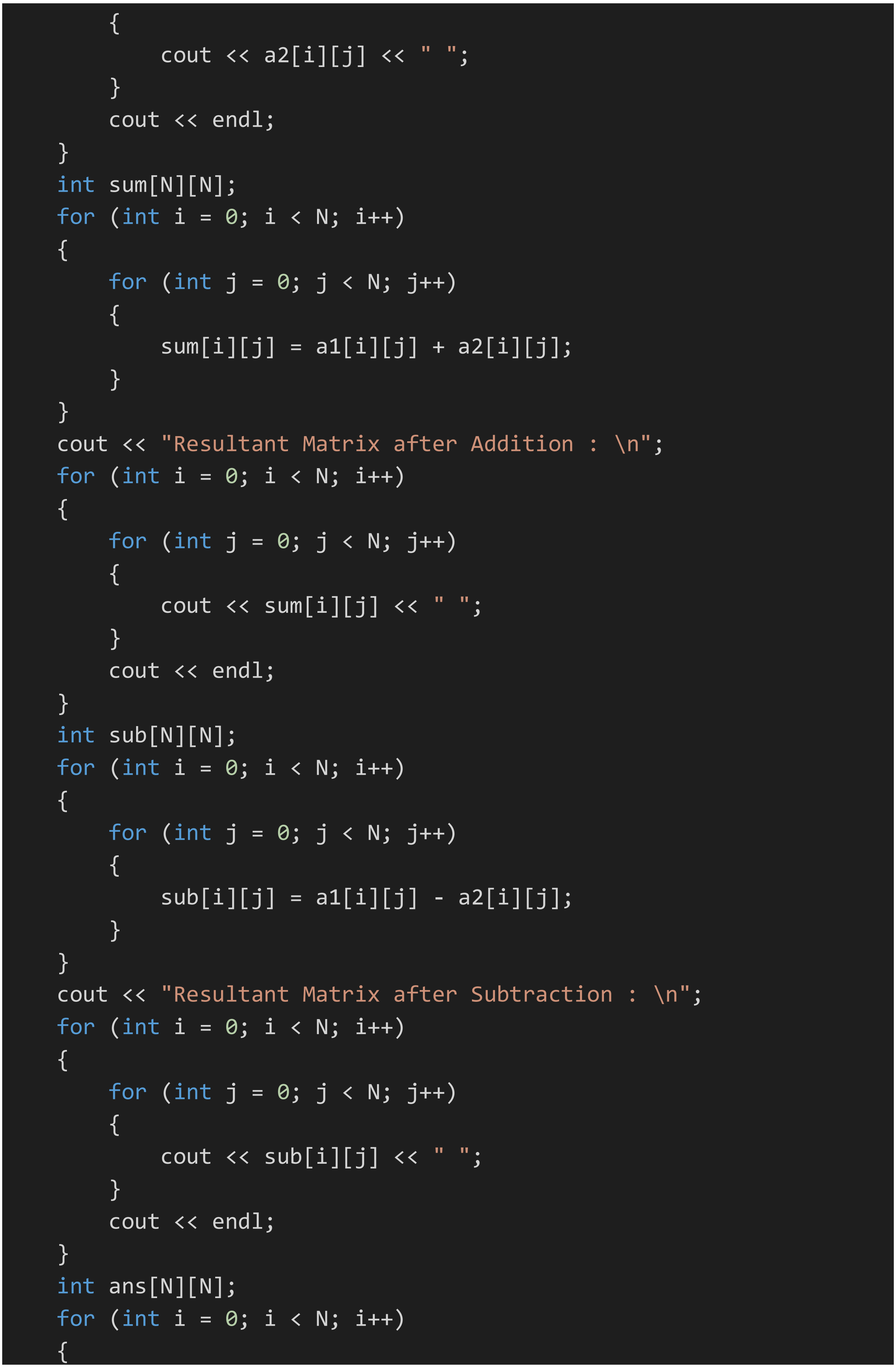
OUTPUT:



```
Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\DELL> cd "g:\my codes\" ; if ($?) { g++ OOM.CPP -o OOM } ; if ($?) { .\OOM }
Complex Number 2 + 7i created.
Complex Number 2 + 2i created.
Addition of two complex Number created = 4 + 9i
Subtraction of two complex Number created = 0 - 5i
PS G:\my codes>
```

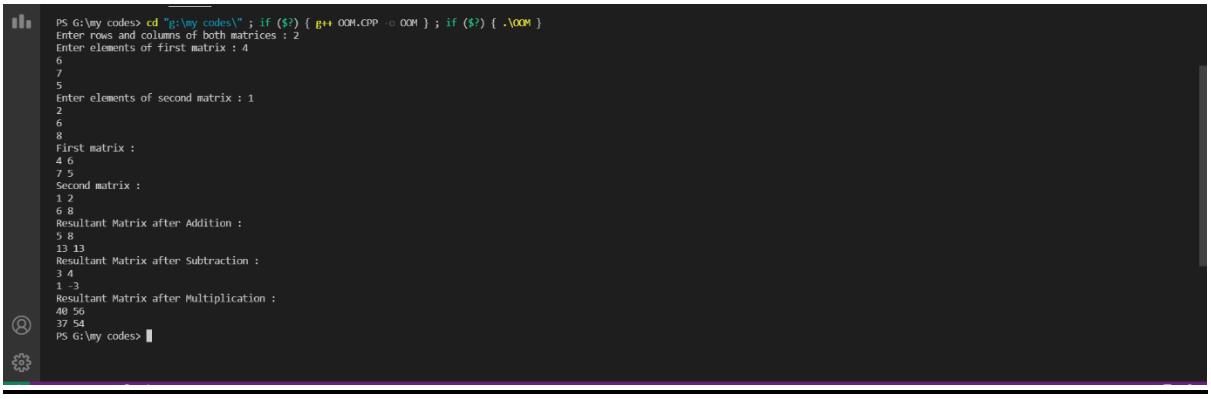
2.Write a program to add, subtract and multiply two matrices.

```
#include <bits/stdc++.h>
using namespace std;

int main()
{
    int N;
    cout << "Enter rows and columns of both matrices : ";
    cin >> N;
    int a1[N][N];
    int a2[N][N];
    cout << "Enter elements of first matrix : ";
    for (int i = 0; i < N; i++)
    {
        for (int j = 0; j < N; j++)
        {
            cin >> a1[i][j];
        }
    }
    cout << "Enter elements of second matrix : ";
    for (int i = 0; i < N; i++)
    {
        for (int j = 0; j < N; j++)
        {
            cin >> a2[i][j];
        }
    }
    cout << "First matrix :\n";
    for (int i = 0; i < N; i++)
    {
        for (int j = 0; j < N; j++)
        {
            cout << a1[i][j] << " ";
        }
        cout << endl;
    }
    cout << "Second matrix :\n";
    for (int i = 0; i < N; i++)
    {
        for (int j = 0; j < N; j++)
```



```
        {
            cout << a2[i][j] << " ";
        }
        cout << endl;
    }
    int sum[N][N];
    for (int i = 0; i < N; i++)
    {
        for (int j = 0; j < N; j++)
        {
            sum[i][j] = a1[i][j] + a2[i][j];
        }
    }
    cout << "Resultant Matrix after Addition : \n";
    for (int i = 0; i < N; i++)
    {
        for (int j = 0; j < N; j++)
        {
            cout << sum[i][j] << " ";
        }
        cout << endl;
    }
    int sub[N][N];
    for (int i = 0; i < N; i++)
    {
        for (int j = 0; j < N; j++)
        {
            sub[i][j] = a1[i][j] - a2[i][j];
        }
    }
    cout << "Resultant Matrix after Subtraction : \n";
    for (int i = 0; i < N; i++)
    {
        for (int j = 0; j < N; j++)
        {
            cout << sub[i][j] << " ";
        }
        cout << endl;
    }
    int ans[N][N];
    for (int i = 0; i < N; i++)
    {
        for (int j = 0; j < N; j++)
        {
            ans[i][j] = 0;
        }
    }
    for (int i = 0; i < N; i++)
    {
        for (int j = 0; j < N; j++)
        {
            for (int k = 0; k < N; k++)
            {
                ans[i][j] += a1[i][k] * a2[k][j];
            }
        }
    }
    cout << "Resultant Matrix after Multiplication : \n";
    for (int i = 0; i < N; i++)
    {
        for (int j = 0; j < N; j++)
        {
            cout << ans[i][j] << " ";
        }
        cout << endl;
    }
    return 0;
}
```

<u>OUTPUT:</u>

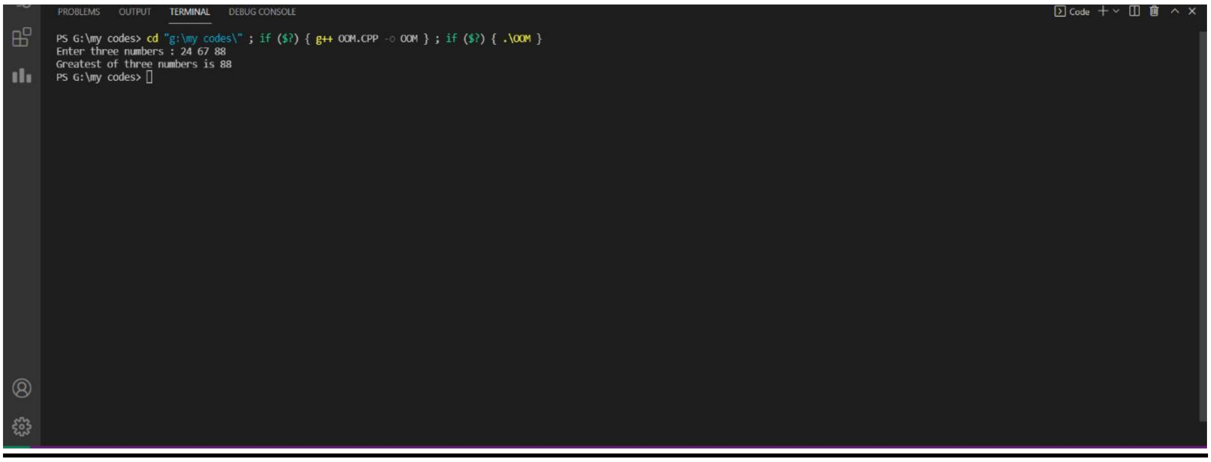


```
PS G:\my codes> cd "g:\my codes\" ; if ($?) { g++ OOM.CPP -o OOM } ; if ($?) { .\OOM }
Enter rows and columns of both matrices : 2
Enter elements of first matrix : 4
6
7
5
Enter elements of second matrix : 1
2
6
8
First matrix :
4 6
7 5
Second matrix :
1 2
6 8
Resultant Matrix after Addition :
5 8
13 13
Resultant Matrix after Subtraction :
3 4
1 -3
Resultant Matrix after Multiplication :
40 56
37 54
PS G:\my codes> █
```

3. Write a program to find greatest of three numbers.

```
#include <bits/stdc++.h>
using namespace std;

void greatestOfThreeNumbers(int a, int b, int c)
{
    if (a >= b && a >= c)
        {
            if (a > b && a > c)
            {
                cout << "Greatest of three numbers is " << a << endl;
            }
            else
            {
                if (a == b && a > c)
                {
                    cout << "A and B both are greatest numbers which is equal to " << a << endl;
                }
                else if (a == b && a == c)
                {
                    cout << "A , B and C all are equal to each other and their value is " << a << endl;
                }
                else
                {
                    cout << "A and C both are greatest numbers which is equal to " << c << endl;
                }
            }
        }
        else if (b >= a && b >= c)
        {
            if (b > c && b > a)
            {
                cout << "Greatest of three numbers is " << b;
            }
            else
            {
                if (b == a && b > c)
                {
                    cout << "A and B both are greatest numbers which is equal to " << a << endl;
                }
                else
                {
                    cout << "B and C both are greatest numbers which is equal to " << c << endl;
                }
            }
        }
        else
        {
            if (c > a && c > b)
            {
                cout << "Greatest of three numbers is " << c << endl;
            }
            else
            {
                if (c == a && c > b)
                {
                    cout << "A and C both are greatest numbers which is equal to " << c << endl;
                }
                else
                {
                    cout << "B and C both are greatest numbers which is equal to " << c << endl;
                }
            }
        }
    }

    int main()
    {
        int a, b, c;
        cout << "Enter three numbers : ";
        cin >> a >> b >> c;
        greatestOfThreeNumbers(a, b, c);
        return 0;
    }

```

OUTPUT:

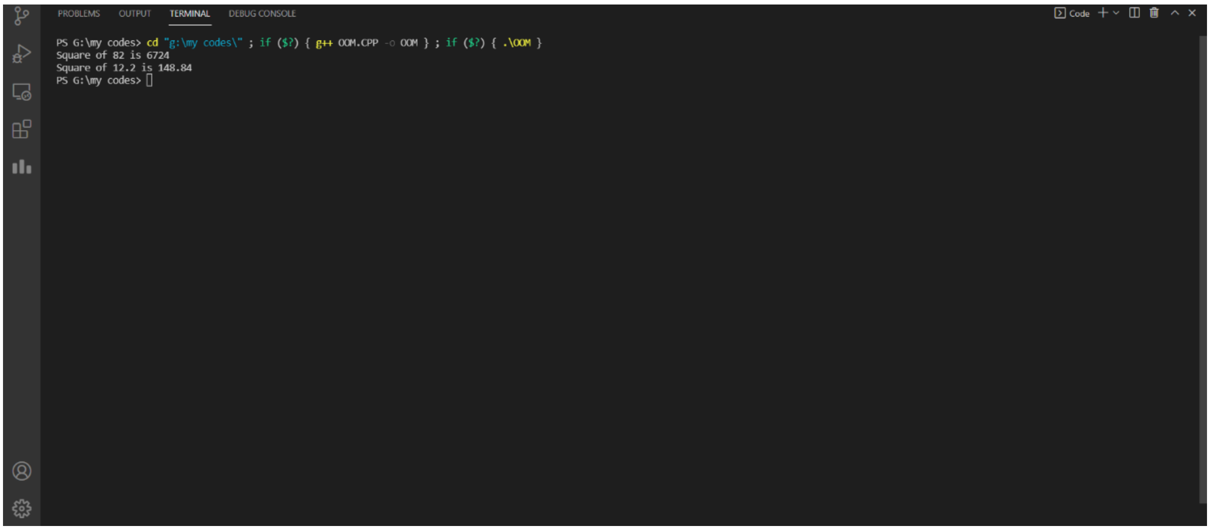


```
PS G:\my codes> cd "g:\my codes\" ; if ($?) { g++ OOM.CPP -o OOM } ; if ($?) { .\OOM }
Enter three numbers : 24 67 88
Greatest of three numbers is 88
PS G:\my codes> 
```

4. Write a program to find out square of given numbers of different data types.

```cpp
#include <bits/stdc++.h>
using namespace std;

template <class T>
void squareNumber(T val)
     {
         T result = val * val;
         cout << "Square of " << val << " is " << result << endl;
     }

     int main()
     {
         squareNumber<int>(82);
         squareNumber<float>(12.2);
         return 0;
     }

```

OUTPUT:

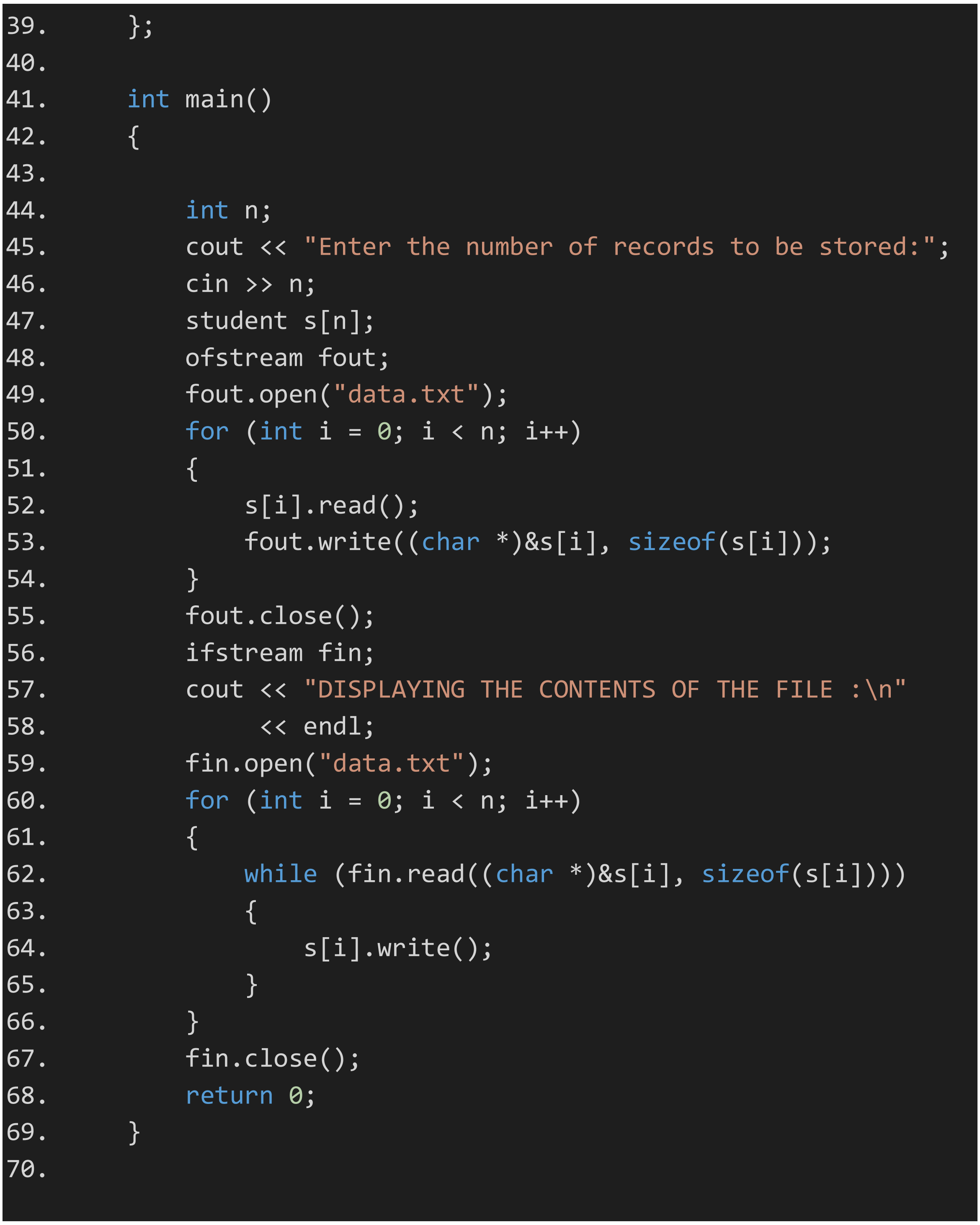


```
PS G:\my codes> cd "g:\my codes\" ; if ($?) { g++ OOM.CPP -o OOM } ; if ($?) { .\OOM }
Square of 82 is 6724
Square of 12.2 is 148.84
PS G:\my codes> 
```

Lab 2

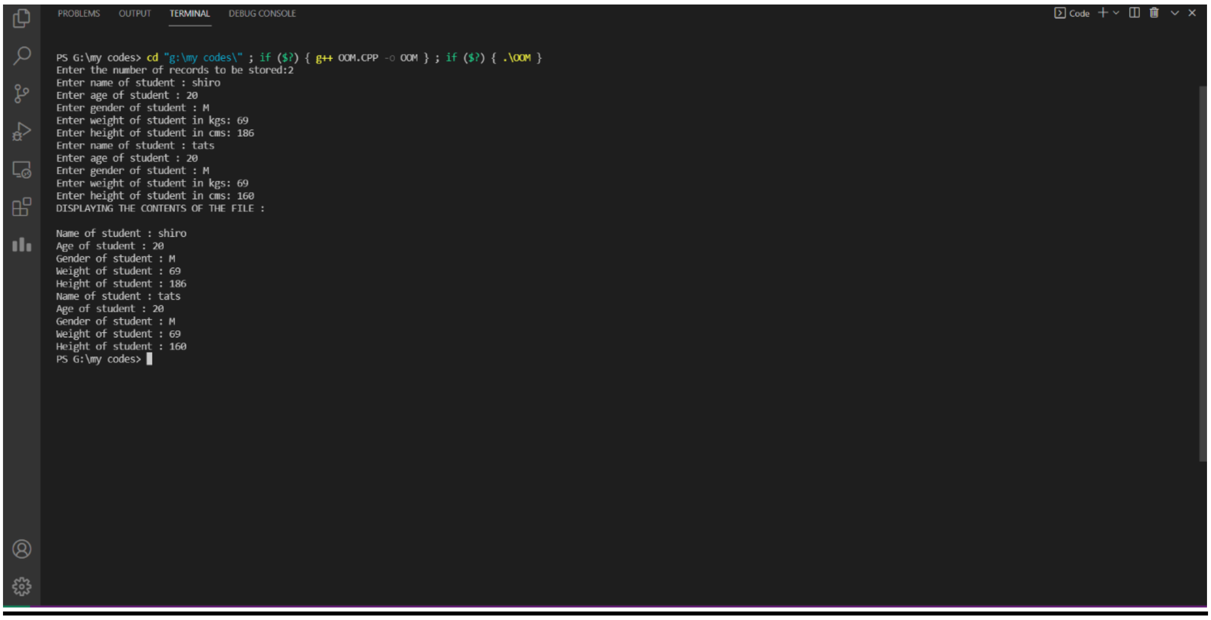
5. **Write a program to read class student info such as name, age, gender, height and weight from the keyboard and to store them on a specified file using read() and write() functions. Again, the same file is opened for reading and displaying the contents of the file on the screen.**

```
#include <bits/stdc++.h>
using namespace std;

class student
    {
        string name;
        int age;
        string gender;
        float weight;
        float height;

    public:
        void read()
        {
            cout << "Enter name of student : ";
            cin >> name;
            cout << "Enter age of student : ";
            cin >> age;
            cout << "Enter gender of student : ";
            cin >> gender;
            cout << "Enter weight of student in kgs: ";
            cin >> weight;
            cout << "Enter height of student in cms: ";
            cin >> height;
        };
        void write()
        {
            cout << "Name of student : " << name << endl;
            cout << "Age of student : " << age << endl;
            cout << "Gender of student : " << gender << endl;
            cout << "Weight of student : " << weight << endl;
            cout << "Height of student : " << height << endl;
        };
```



```cpp
};

int main()
{

    int n;
    cout << "Enter the number of records to be stored:";
    cin >> n;
    student s[n];
    ofstream fout;
    fout.open("data.txt");
    for (int i = 0; i < n; i++)
    {
        s[i].read();
        fout.write((char *)&s[i], sizeof(s[i]));
    }
    fout.close();
    ifstream fin;
    cout << "DISPLAYING THE CONTENTS OF THE FILE :\n"
         << endl;
    fin.open("data.txt");
    for (int i = 0; i < n; i++)
    {
        while (fin.read((char *)&s[i], sizeof(s[i])))
        {
            s[i].write();
        }
    }
    fin.close();
    return 0;
}

```

OUTPUT:



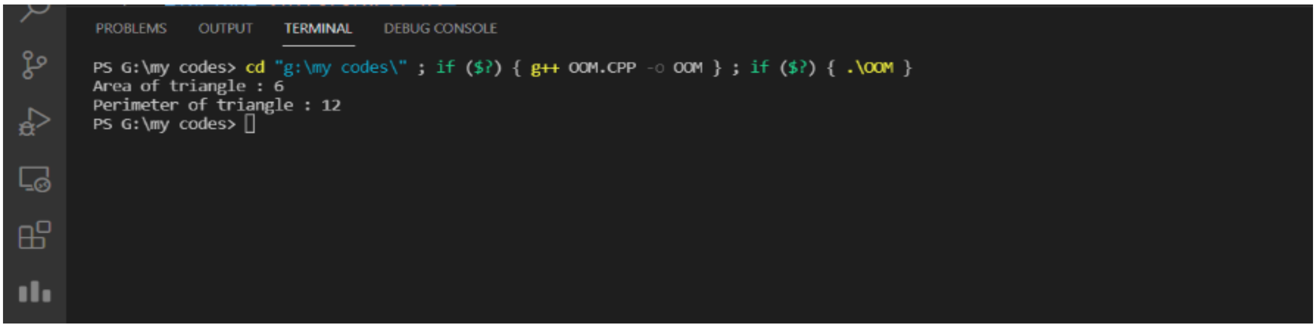
```
PS G:\my codes> cd "g:\my codes\" ; if ($?) { g++ OOM.CPP -o OOM } ; if ($?) { .\OOM }
Enter the number of records to be stored:2
Enter name of student : shiro
Enter age of student : 20
Enter gender of student : M
Enter weight of student in kgs: 69
Enter height of student in cms: 186
Enter name of student : tats
Enter age of student : 20
Enter gender of student : M
Enter weight of student in kgs: 69
Enter height of student in cms: 160
DISPLAYING THE CONTENTS OF THE FILE :

Name of student : shiro
Age of student : 20
Gender of student : M
Weight of student : 69
Height of student : 186
Name of student : tats
Age of student : 20
Gender of student : M
Weight of student : 69
Height of student : 160
PS G:\my codes> 
```

Lab 3

6. **Write a program to print the area and perimeter of a triangle, having sides of 3, 4 and 5 units by creating a class named ‘Triangle’, with a function to print the area and perimeter.**

```cpp
#include <bits/stdc++.h>
using namespace std;

    class Triangle
    {

        float a, b, c;

    public:
        void area(float a, float b, float c)
        {
            float s = (a + b + c) / 2;
            float area;
            area = sqrt(s * (s - a) * (s - b) * (s - c));
            cout << "Area of triangle : " << area << endl;
        }
        void perimeter(float a, float b, float c)
        {
            float perimeter;
            perimeter = a + b + c;
            cout << "Perimeter of triangle : " << perimeter << endl;
        }
    };

    int main()
    {
        Triangle T;
        T.area(3, 4, 5);
        T.perimeter(3, 4, 5);
        return 0;
    }

```

OUTPUT:



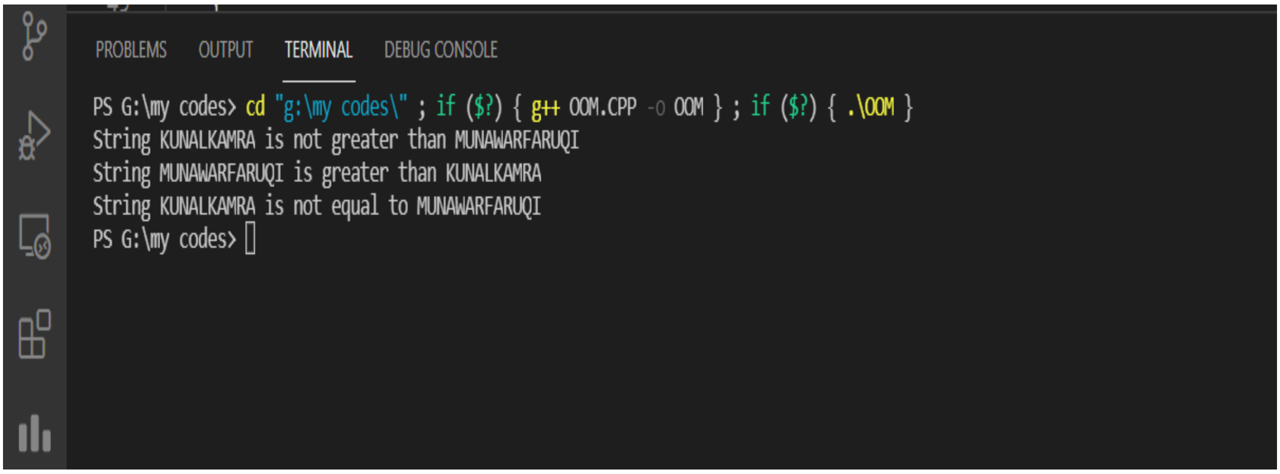
```
PS G:\my codes> cd "g:\my codes\" ; if ($?) { g++ OOM.CPP -o OOM } ; if ($?) { .\OOM }
Area of triangle : 6
Perimeter of triangle : 12
PS G:\my codes> 
```

7. **Write a C++ program to compare two strings using overloading.**

```
#include <bits/stdc++.h>
using namespace std;

    class String
    {
    public:
        string s;
        void operator=(String s1)
        {
            if (s.size() == s1.s.size())
            {
                int count = 0;
                for (int i = 0; i < s1.s.size(); i++)
                {
                    if (s[i] == s1.s[i])
                    {
                        count++;
                    }
                    else
                    {
                        cout << "String " << s << " is not equal to " << s1.s << endl;
                        break;
                    }
                }

                if (count == s.size())
                {
                    cout << "String " << s << " is equal to " << s1.s << endl;
                }
            }
            else
            {
                cout << "String " << s << " is not equal to " << s1.s << endl;
            }
        }

```

```cpp
        void operator>(String s1)
        {
            if (s.size() > s1.s.size())
            {
                cout << "String " << s << " is greater than " << s1.s << endl;
            }
            else
            {
                cout << "String " << s << " is not greater than " << s1.s << endl;
            }
        }
        void operator<(String s1)
        {
            if (s.size() < s1.s.size())
            {
                cout << "String " << s1.s << " is greater than " << s << endl;
            }
            else
            {
                cout << "String " << s1.s << " is not greater than " << s << endl;
            }
        }
    };

    int main()
    {
        String s1, s2;
        s1.s = "KUNALKAMRA";
        s2.s = "MUNAWARFARUQI";
        s1 > s2;
        s1 < s2;
        s1 = s2;
        return 0;
    }

```

OUTPUT:

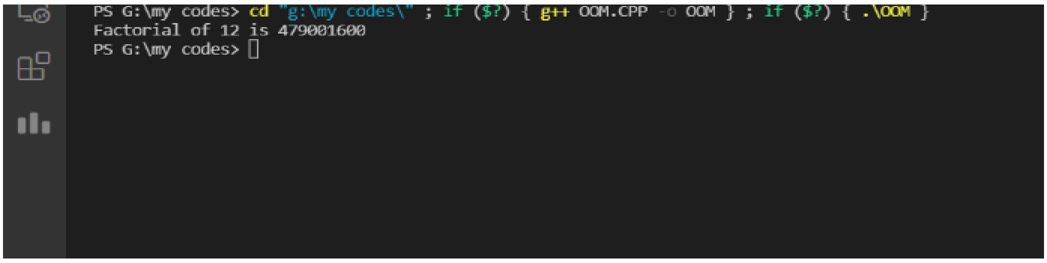


```
PS G:\my codes> cd "g:\my codes\" ; if ($?) { g++ OOM.CPP -o OOM } ; if ($?) { .\OOM }
String KUNALKAMRA is not greater than MUNAWARFARUQI
String MUNAWARFARUQI is greater than KUNALKAMRA
String KUNALKAMRA is not equal to MUNAWARFARUQI
PS G:\my codes> 
```

8. Write a C++ program to find the factorial of a number using class and function declared outside the class.

```
#include <bits/stdc++.h>
    using namespace std;

    class Factorial
    {
        int n;

    public:
        void factorial(int n);
    };

    void Factorial ::factorial(int n)
    {
        int fact = 1;
        for (int i = n; i > 0; i--)
        {
            fact *= i;
        }
        cout << "Factorial of " << n << " is " << fact << endl;
    }

    int main()
    {
        Factorial f;
        f.factorial(12);
        return 0;
    }
```

OUTPUT:



```
PS G:\my codes> cd "g:\my codes\" ; if ($?) { g++ OOM.CPP -o OOM } ; if ($?) { .\OOM }
Factorial of 12 is 479001600
PS G:\my codes> 
```

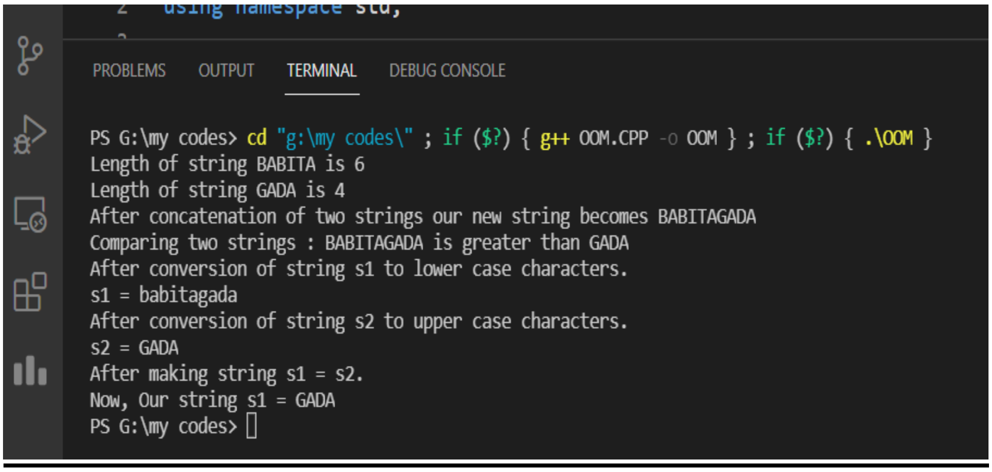
Lab 4

9. **Implement the class string containing the following functions:**

- **Overload + operator to carry out the concatenation of strings**
- **Overload = operator to carry out string copy**
- **Overload <= operator to carry out comparison of strings**
- **Function to display the length of the string**
- **Function tolower() to convert uppercase letter to lowercase**
- **Function toupper() to convert lowercase letter to uppercase**

```
#include <bits/stdc++.h>
using namespace std;

class String
{
public:
    string s;
    void operator+(String S)
    {
        s = s + S.s;
        cout << s << endl;
    }
    void operator=(String S)
    {
        s = S.s;
    }
    void operator<=(String S)
    {
        if (S.s.size() > s.size())
        {
            cout << S.s << " is greater than " << s << endl;
        }
        else if (S.s.size() == s.size())
        {
            int count = 0;
            for (int i = 0; i < S.s.size(); i++)
            {
                if (s[i] == S.s[i])
                {
                    count++;
                }
                else
                {
                    cout << "String " << s << " is not equal to " << S.s << endl;
                    break;
                }
            }

            if (count == s.size())
            {
                cout << "String " << s << " is equal to " << S.s << endl;
            }
        }
        else
        {
            cout << S.s << " is neither equal to nor greater than " << s << endl;
        }
    }
    void length()
    {
        cout << "Length of string " << s << " is " << s.size() << endl;
    }
    void toLower()
    {
        for (int i = 0; i <= s.size(); i++)
        {
            if (isupper(s[i]))
            {
                s[i] += 32;
            }
        }
    }
    void toUpper()
    {
        for (int i = 0; i < s.size(); i++)
        {
            if (islower(s[i]))
                s[i] -= 32;
        }
    }
    void display()
    {
        cout << s << endl;
    }
};

int main()
{
    String s1, s2;
    s1.s = "BABITA";
    s2.s = "GADA";
    s1.length();
    s2.length();
    cout << "After concatenation of two strings our new string becomes ";
    s1 + s2;
    cout << "Comparing two strings : ";
    s2 <= s1;
    s1.toLower();
    cout << "After conversion of string s1 to lower case characters.\ns1 = ";
    s1.display();
    s2.toUpper();
    cout << "After conversion of string s2 to upper case characters.\ns2 = ";
    s2.display();
    cout << "After making string s1 = s2.\nNow, Our string s1 = ";
    s1 = s2;
    s1.display();
    return 0;
}
```

OUTPUT:



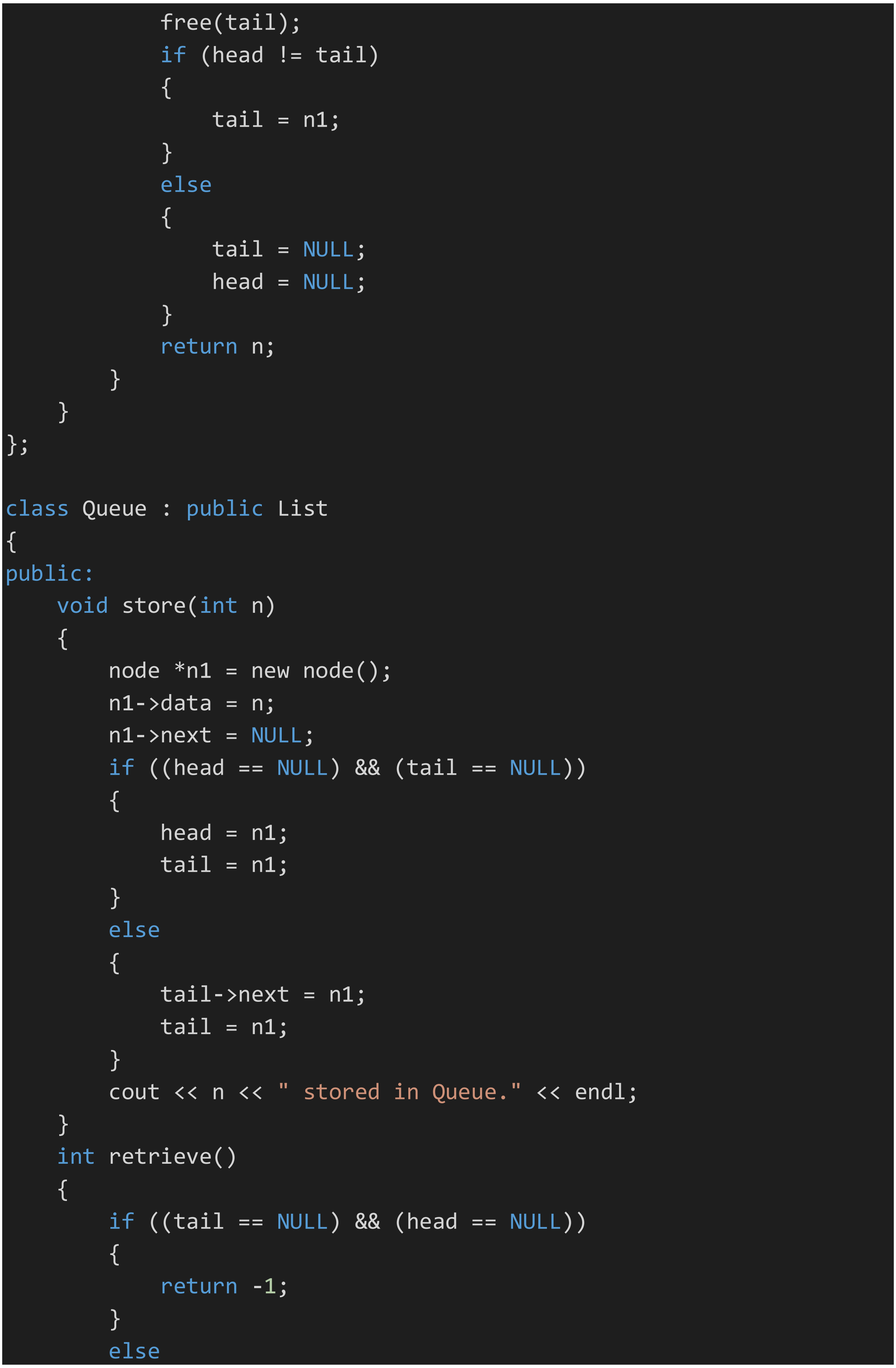
```
using namespace std;
```

```
PS G:\my codes> cd "g:\my codes\" ; if ($?) { g++ OOM.CPP -o OOM } ; if ($?) { .\OOM }
Length of string BABITA is 6
Length of string GADA is 4
After concatenation of two strings our new string becomes BABITAGADA
Comparing two strings : BABITAGADA is greater than GADA
After conversion of string s1 to lower case characters.
s1 = babitagada
After conversion of string s2 to upper case characters.
s2 = GADA
After making string s1 = s2.
Now, Our string s1 = GADA
PS G:\my codes> 
```

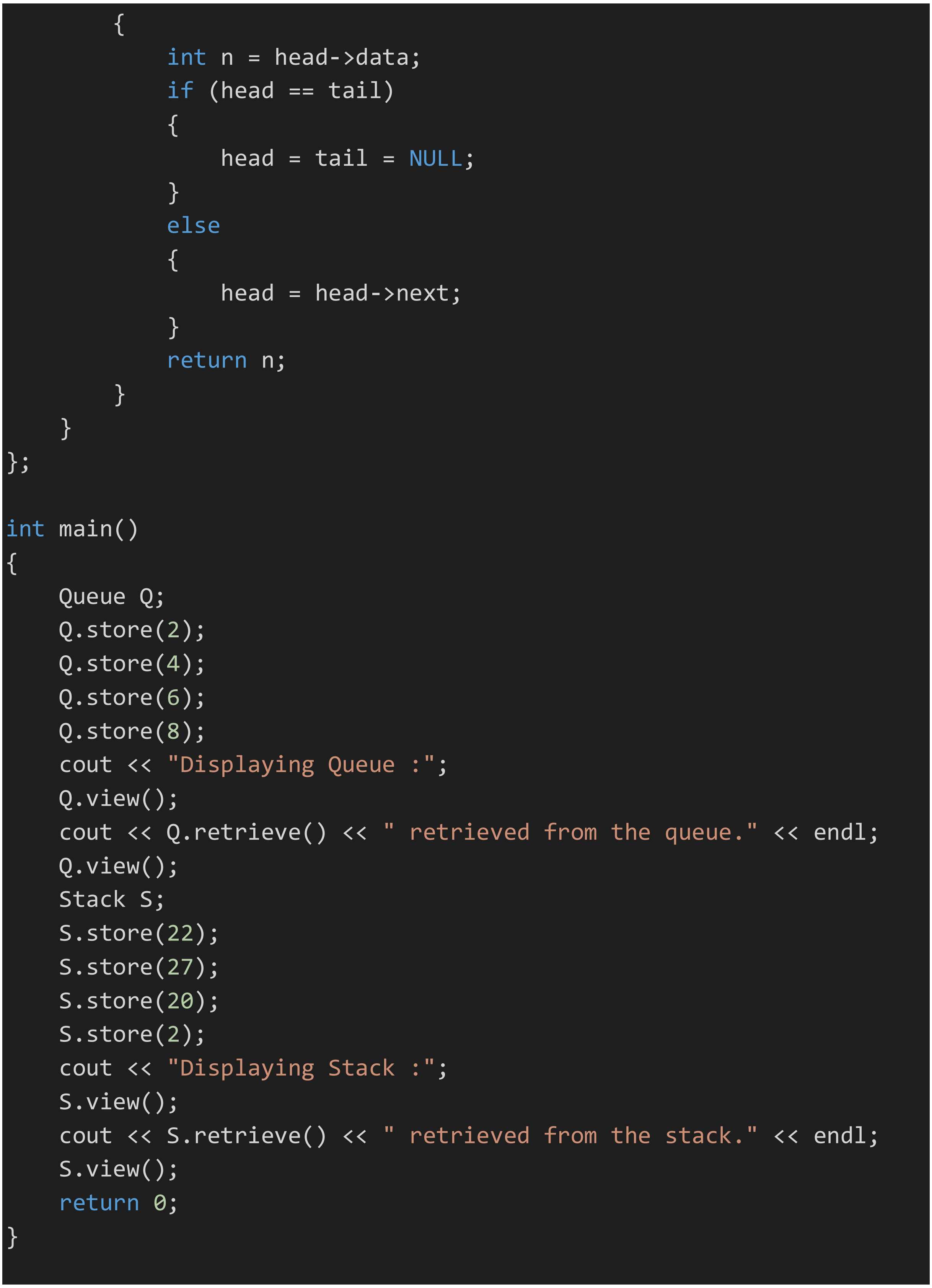
10. Create a class called LIST with functions store() and retrieve(). To store a value, call store function and to retrieve a value, call retrieve function. Derive two classes stack and queue from it and override store and retrieve.

```
#include <bits/stdc++.h>
using namespace std;

struct node
{
    int data;
    node *next;
};

class List
{
public:
    node *head = NULL, *tail = NULL;
    void view()
    {
        node *n = head;
        if (head == NULL)
        {
            cout << "\n No elements found...";
        }
        else
        {
            cout << " ";
            while (n != NULL)
            {
                if (n->next == NULL)
                {
                    cout << n->data;
                }
                else
                {
                    cout << n->data << "->";
                }
                n = n->next;
            }
```

```cpp
            cout << endl;
        }
    }
    virtual void store(int n) = 0;
    virtual int retrieve() = 0;
};

class Stack : public List
{
public:
    void store(int n)
    {
        node *n1 = new node();
        n1->data = n;
        n1->next = NULL;
        if ((head == NULL) && (tail == NULL))
        {
            head = n1;
            tail = n1;
        }
        else
        {
            tail->next = n1;
            tail = n1;
        }
        cout << n << " stored in Stack." << endl;
    }
    int retrieve()
    {
        if ((tail == NULL) && (head == NULL))
        {
            return -1;
        }
        else
        {
            int n = tail->data;
            node *n1 = head;
            while ((n1->next != tail) && (head != tail))
            {
                n1 = n1->next;
            }
            n1->next = NULL;
```

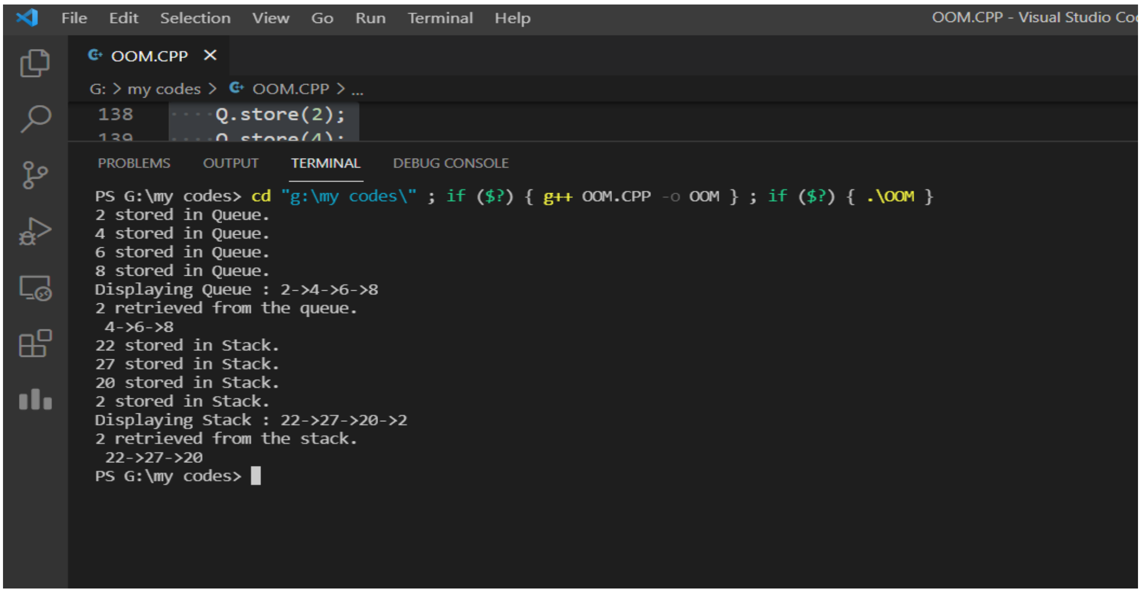


```
            free(tail);
            if (head != tail)
            {
                tail = n1;
            }
            else
            {
                tail = NULL;
                head = NULL;
            }
            return n;
        }
    }
};

class Queue : public List
{
public:
    void store(int n)
    {
        node *n1 = new node();
        n1->data = n;
        n1->next = NULL;
        if ((head == NULL) && (tail == NULL))
        {
            head = n1;
            tail = n1;
        }
        else
        {
            tail->next = n1;
            tail = n1;
        }
        cout << n << " stored in Queue." << endl;
    }
    int retrieve()
    {
        if ((tail == NULL) && (head == NULL))
        {
            return -1;
        }
        else
```



```cpp
        {
            int n = head->data;
            if (head == tail)
            {
                head = tail = NULL;
            }
            else
            {
                head = head->next;
            }
            return n;
        }
    }
};

int main()
{
    Queue Q;
    Q.store(2);
    Q.store(4);
    Q.store(6);
    Q.store(8);
    cout << "Displaying Queue :";
    Q.view();
    cout << Q.retrieve() << " retrieved from the queue." << endl;
    Q.view();
    Stack S;
    S.store(22);
    S.store(27);
    S.store(20);
    S.store(2);
    cout << "Displaying Stack :";
    S.view();
    cout << S.retrieve() << " retrieved from the stack." << endl;
    S.view();
    return 0;
}
```

OUTPUT:



```
    Q.store(2);
    Q.store(4);
```

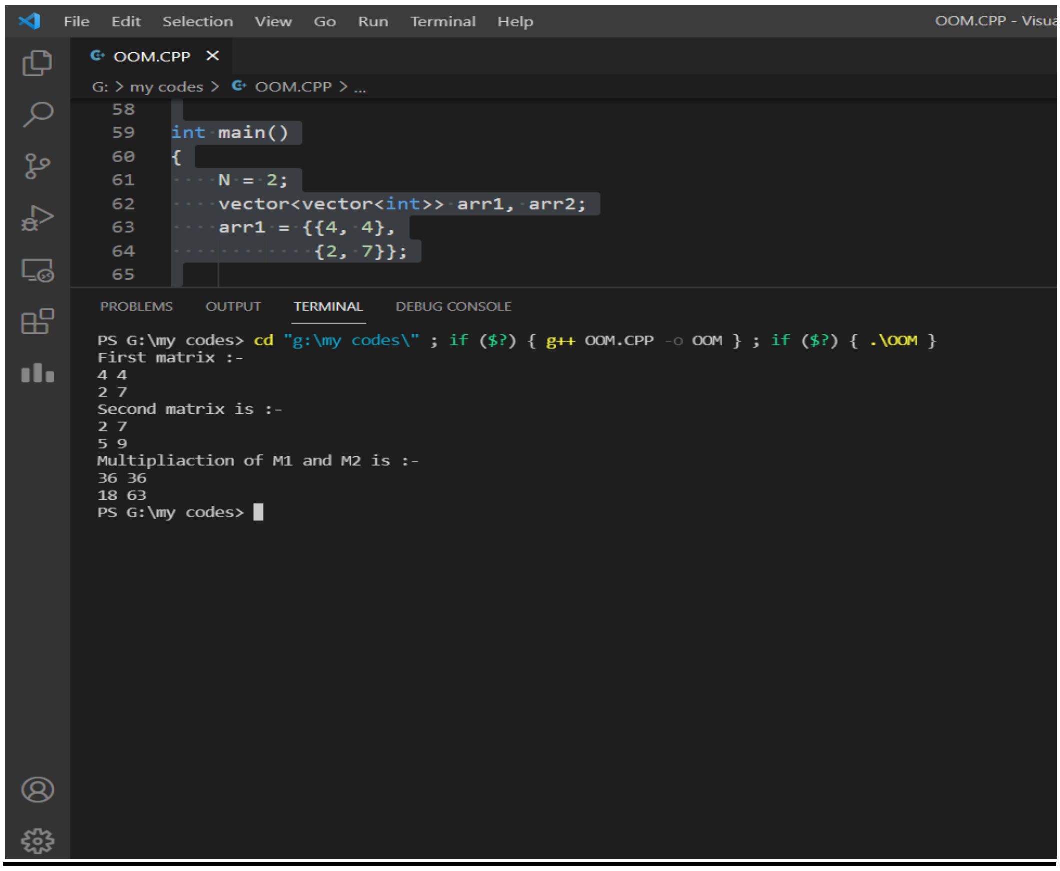
```
PS G:\my codes> cd "g:\my codes\" ; if ($?) { g++ OOM.CPP -o OOM } ; if ($?) { .\OOM }
2 stored in Queue.
4 stored in Queue.
6 stored in Queue.
8 stored in Queue.
Displaying Queue : 2->4->6->8
2 retrieved from the queue.
 4->6->8
22 stored in Stack.
27 stored in Stack.
20 stored in Stack.
2 stored in Stack.
Displaying Stack : 22->27->20->2
2 retrieved from the stack.
 22->27->20
PS G:\my codes> 
```

Lab 5

11. Write a program for multiplication of two matrices using OOP.

```
#include <bits/stdc++.h>
using namespace std;
#define rows 50
#define cols 50

int N;
class Matrix
{
    int mat[rows][cols];

public:
    void setMatrix(vector<vector<int>> &A)
    {
        for (int i = 0; i < N; i++)
        {
            for (int j = 0; j < N; j++)
            {
                mat[i][j] = A[i][j];
            }
        }
    }
    void display()
    {
        for (int i = 0; i < N; i++)
        {
            for (int j = 0; j < N; j++)
            {
                cout << mat[i][j] << " ";
            }
            cout << endl;
        }
    }

    void operator*(Matrix m)
    {
        int prod[N][N];
        for (int i = 0; i < N; i++)
        {
            for (int j = 0; j < N; j++)
            {
                prod[i][j] = 0;
                for (int k = 0; k < N; k++)
                {
                    prod[i][k] += mat[i][k] * (m.mat[k][j]);
                }
            }
        }
        for (int i = 0; i < N; i++)
        {
            for (int j = 0; j < N; j++)
            {
                cout << prod[i][j] << " ";
            }
            cout << endl;
        }
    }
};

int main()
{
    N = 2;
    vector<vector<int>> arr1, arr2;
    arr1 = {{4, 4},
            {2, 7}};

    arr2 = {{2, 7},
            {5, 9}};

    Matrix M1, M2;
    M1.setMatrix(arr1);
    M2.setMatrix(arr2);
    cout << "First matrix :- " << endl;
    M1.display();
    cout << "Second matrix is :- " << endl;
    M2.display();
    cout << "Multipliaction of M1 and M2 is :-" << endl;
    M1 *M2;
    return 0;
}
```

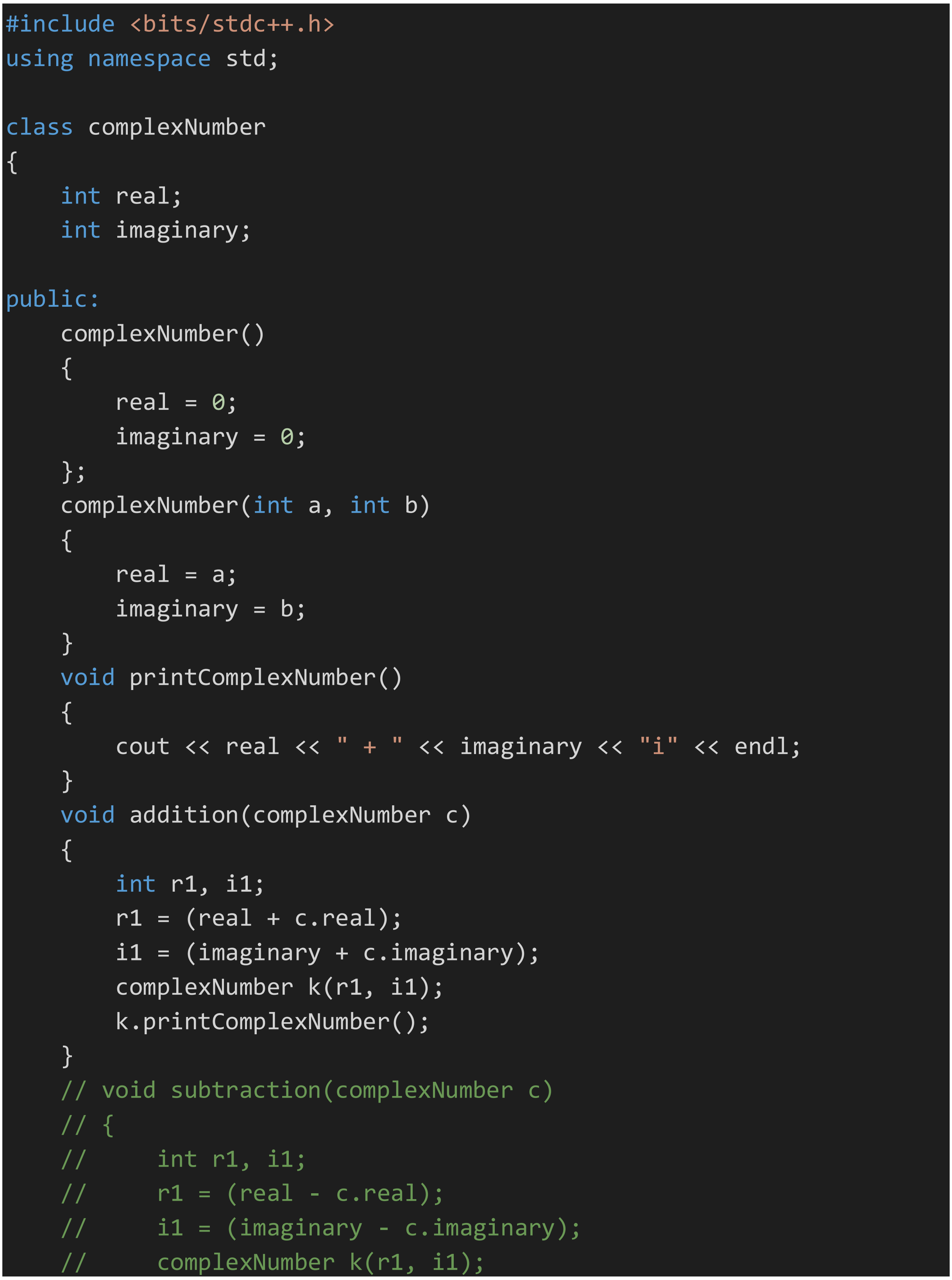
OUTPUT:



```
int main()
{
    N = 2;
    vector<vector<int>> arr1, arr2;
    arr1 = {{4, 4},
            {2, 7}};
```

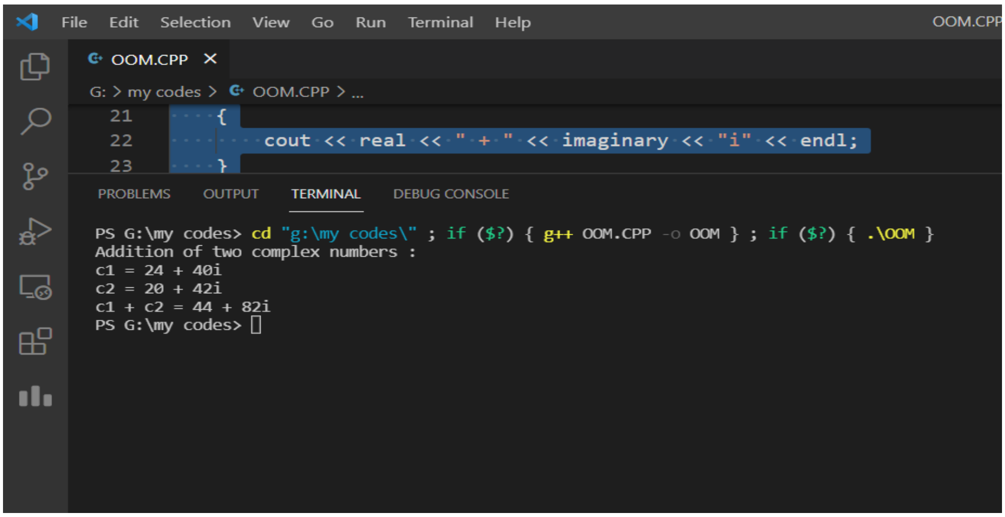
```
PS G:\my codes> cd "g:\my codes\" ; if ($?) { g++ OOM.CPP -o OOM } ; if ($?) { .\OOM }
First matrix :-
4 4
2 7
Second matrix is :-
2 7
5 9
Multipliaction of M1 and M2 is :-
36 36
18 63
PS G:\my codes> 
```

12. Write a program to perform addition of two complex numbers using OOP.



```cpp
#include <bits/stdc++.h>
using namespace std;

class complexNumber
{
    int real;
    int imaginary;

public:
    complexNumber()
    {
        real = 0;
        imaginary = 0;
    };
    complexNumber(int a, int b)
    {
        real = a;
        imaginary = b;
    }
    void printComplexNumber()
    {
        cout << real << " + " << imaginary << "i" << endl;
    }
    void addition(complexNumber c)
    {
        int r1, i1;
        r1 = (real + c.real);
        i1 = (imaginary + c.imaginary);
        complexNumber k(r1, i1);
        k.printComplexNumber();
    }
    // void subtraction(complexNumber c)
    // {
    //     int r1, i1;
    //     r1 = (real - c.real);
    //     i1 = (imaginary - c.imaginary);
    //     complexNumber k(r1, i1);
    //      k.printComplexNumber();
    // }
};

int main()
{
    complexNumber c1(24, 40), c2(20, 42);
    cout << "Addition of two complex numbers :\n";
    cout << "c1 = ";
    c1.printComplexNumber();
    cout << "c2 = ";
    c2.printComplexNumber();
    cout << "c1 + c2 = ";
    c1.addition(c2);
    // c1.subtraction(c2);
    return 0;
}
```

OUTPUT:



```cpp
    {
        cout << real << " + " << imaginary << "i" << endl;
    }
```

```
PS G:\my codes> cd "g:\my codes\" ; if ($?) { g++ OOM.CPP -o OOM } ; if ($?) { .\OOM }
Addition of two complex numbers :
c1 = 24 + 40i
c2 = 20 + 42i
c1 + c2 = 44 + 82i
PS G:\my codes> 
```