

**DSA (Data Structure and Algorithms) Lab**

**Assignment # 1**

**Semester**: 3rdSemester

**Section**: C

**Submitted To:**

**Submitted By:**

**Name**: Muhammad Afzal

**Roll No**: 22-CS-035

**Task 1:**

**Code:**

#include <iostream>

using namespace std;

int main()

{

    system("cls");

    int size;

    cout << "Enter the size of Array : ";

    cin >> size;

    int \*A = new int[size];

    for (int i = 0; i < size; i++)

    {

        cout << "\nEntre the Value for the Array at index " << i << " : ";

        cin >> A[i];

    }

    for (int i = 0; i < size - 1; i++)

    {

        int temp1;

        for (int j = i + 1; j < size; j++)

        {

            if (A[j] < A[i])

            {

                temp1 = A[j];

                A[j] = A[i];

                A[i] = temp1;

            }

        }

    }

    int median = (size % 2 != 0) ? size / 2 : -1;

    if (median == -1)

    {

        median = (A[size / 2] + A[(size / 2) - 1]) / 2;

    }

    else

    {

        median = A[median];

    }

    cout << "\n\nThe Array is : { ";

    for (int i = 0; i < size; i++)

    {

        if (i == size - 1)

        {

            cout << A[i] << " }\n\n"

                 << endl;

        }

        else

        {

            cout << A[i] << " , ";

        }

    }

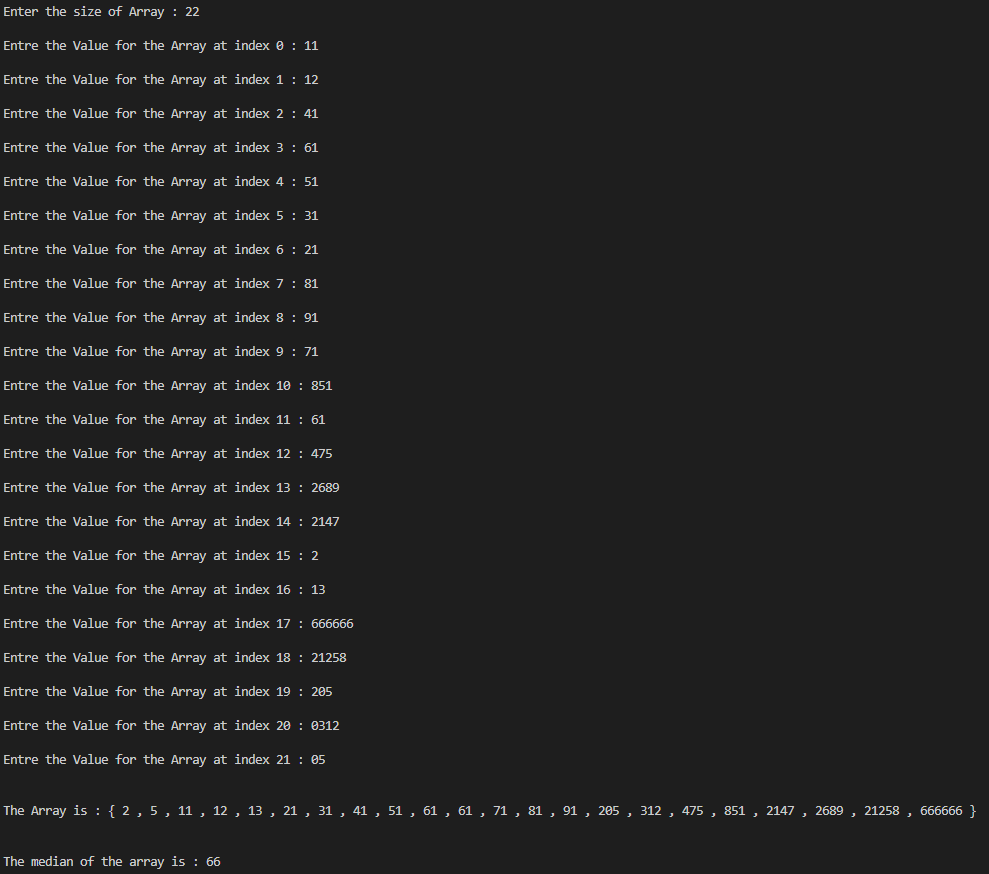
    cout << "The median of the array is : " << median << endl

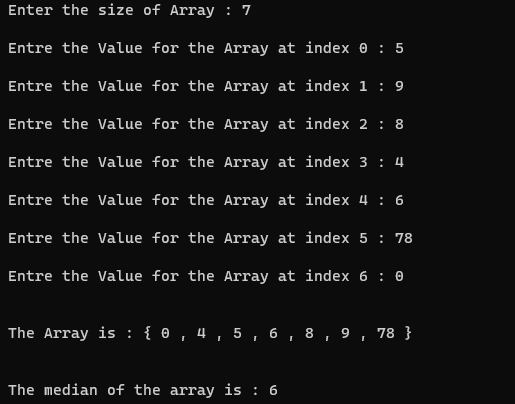
         << endl;

    return 0;

}

**Output:**

****

****

**Task 2:**

**Code:**

#include <iostream>

using namespace std;

class Matrix

{

private:

    int \*\*A;

    int size;

    string name;

public:

    Matrix(int size = 3)

    {

        this->name = name;

        this->size = size;

        A = new int \*[size];

        for (int i = 0; i < size; i++)

        {

            A[i] = new int[size];

        }

        for (int i = 0; i < size; i++)

        {

            for (int j = 0; j < size; j++)

            {

                A[j][i] = 0;

            }

        }

    }

    void setdata(int i, int j, int value)

    {

        A[i][j] = value;

    }

    void getdata()

    {

        cout << "Enter the values for the matrix : " << endl;

        for (int i = 0; i < this->size; i++)

        {

            cout << "Column " << i + 1 << " : " << endl;

            for (int j = 0; j < this->size; j++)

            {

                cout << "Row " << j + 1 << " : ";

                std::cin >> A[j][i];

            }

            cout << endl;

        }

    }

    void display()

    {

        int i = 0, j = 0;

        cout << "Values for the matrix : \n"

             << endl;

        for (int i = 0; i < this->size; i++)

        {

            cout << "  |";

            for (int j = 0; j < this->size; j++)

            {

                cout << "  ";

                cout << A[i][j];

            }

            cout << "  |";

            cout << endl;

        }

    }

    friend std::ostream &operator<<(std::ostream &o, Matrix const &M)

    {

        int i = 0, j = 0;

        o << "Values for the matrix : \n"

          << endl;

        for (int i = 0; i < M.size; i++)

        {

            o << "  |";

            for (int j = 0; j < M.size; j++)

            {

                o << "  ";

                o << M.A[i][j];

            }

            o << "  |";

            o << endl;

        }

        return o;

    }

    Matrix operator\*(Matrix &M)

    {

        Matrix m;

        for (int i = 0; i < size; i++)

        {

            for (int j = 0; j < size; j++)

            {

                int val = (this->A[i][0] \* M.A[0][j]) + (this->A[i][1] \* M.A[1][j]) + (this->A[i][2] \* M.A[2][j]);

                m.setdata(j, i, val);

            }

        }

        return m;

    }

    Matrix operator-(Matrix &M)

    {

        Matrix m;

        for (int i = 0; i < size; i++)

        {

            for (int j = 0; j < size; j++)

            {

                m.setdata(j, i, (this->A[j][i] - M.A[j][i]));

            }

        }

        return m;

    }

    Matrix operator+(Matrix &M)

    {

        Matrix m;

        for (int i = 0; i < size; i++)

        {

            for (int j = 0; j < size; j++)

            {

                m.setdata(j, i, (this->A[j][i] + M.A[j][i]));

            }

        }

        return m;

    }

    ~Matrix()

    {

        for (int i = 0; i < size; i++)

        {

            delete[] A[i];

        }

        delete[] A;

    }

};

int main()

{

    system("cls");

    Matrix M1, M2;

    int choice = -1;

    while (true)

    {

    check:

        system("cls");

        cout << "\t\t\t  MAIN MENU\n\t\t\t~~~~~~~~~~~~\n";

        cout << "1 - Read Matrix - A" << endl;

        cout << "2 - Read Matrix - B" << endl;

        cout << "3 - Addition of Matrix - A and Matrix - B" << endl;

        cout << "4 - Difference of Matrix - A and Matrix - B" << endl;

        cout << "5 - Product of Matrix - A and Matrix - B" << endl;

        cout << "6 - Exit Program";

        cout << "\n\nYour Choice ---->  ";

        std::cin >> choice;

        switch (choice)

        {

        case 1:

            M1.display();

            cout << "\nIf want to change the matrix values press 1 else 0 : ";

            std::cin >> choice;

            switch (choice)

            {

            case 1:

                M1.getdata();

                break;

            default:

                break;

            }

            break;

        case 2:

            M2.display();

            cout << "\nIf want to change the matrix values press 1 else 0 : ";

            std::cin >> choice;

            switch (choice)

            {

            case 1:

                M2.getdata();

                break;

            }

            break;

        case 3:

            cout << (M1 + M2) << endl

                 << endl;

            break;

        case 4:

            cout << (M1 - M2) << endl

                 << endl;

            break;

        case 5:

            cout << (M1 \* M2) << endl

                 << endl;

            break;

        case 6:

            exit(0);

            break;

        default:

            cout << "Wrong Input : ";

            goto check;

        }

        std::cin.ignore();

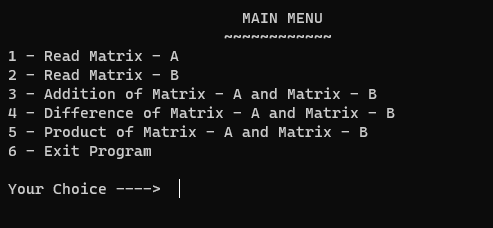
        std::cin.ignore();

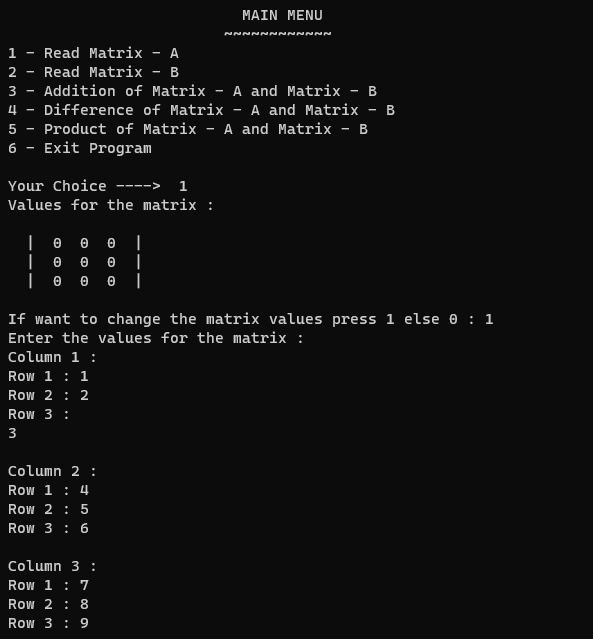
    }

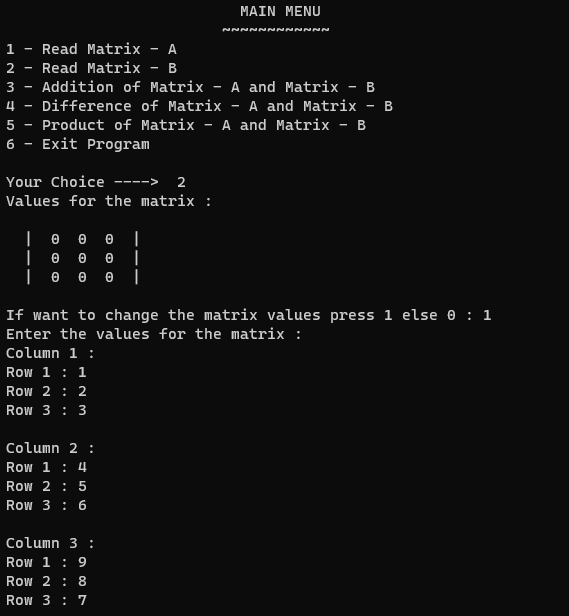
    return 0;

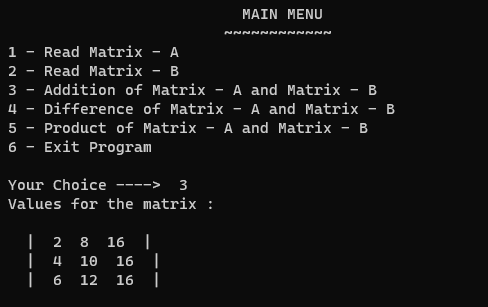
}

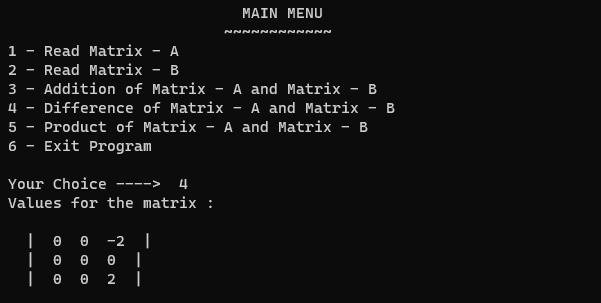
**Output:**

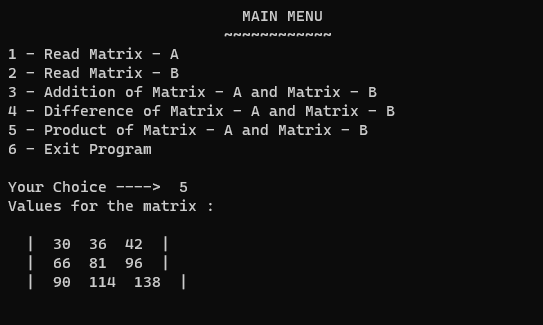
****

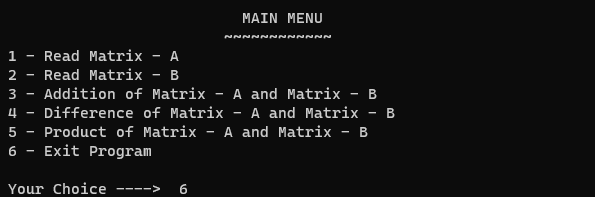
****

****

****

****

****

****