**OOP lab2**

**Name:** Muhammad Raza Mustafa

**Roll number:** 24k-1017 **Section:** BCS-1H

**Task1:**

#include<iostream>

#include<climits>

using namespace std;

int main() {

    int n;

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

    cin >> n;

    int\* arr = new int[n];

    for(int i = 0; i < n; i++) {

        cout << "Enter element " << i + 1 << " : ";

        cin >> arr[i];

    }

    int sum = 0, min = INT\_MAX;

    float avg;

    for(int i = 0; i < n; i++) {

        sum += arr[i];

        if(arr[i] < min) {

            min = arr[i];

        }

    }

    avg = float(sum) / float(n);

    cout << "Sum : " << sum << endl;

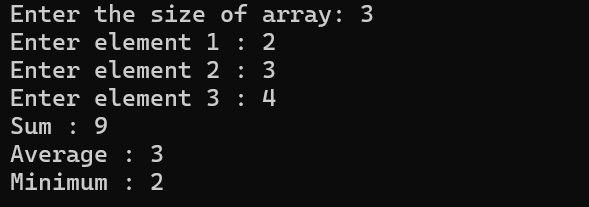
    cout << "Average : " << avg << endl;

    cout << "Minimum : " << min << endl;

    delete[] arr;

    return 0;

}

**Output  
**

**Task2:**

#include<iostream>

using namespace std;

int main() {

    int row1, col1,row2,col2;

    // Dimensions Input

    cout<<"Enter number of rows for matrix1 ";

    cin>>row1;

    cout<<"Enter number of cols for matrix1 ";

    cin>>col1;

    cout<<"Enter number of rows for matrix2 ";

    cin>>row2;

    cout<<"Enter number of cols for matrix2 ";

    cin>>col2;

    //Memory allocation

    int \*\*matrix1 = new int\*[row1];

    for(int i = 0; i < row1; i++) {

        matrix1[i] = new int[col1];

    }

    int \*\*matrix2 = new int\*[row2];

    for(int i = 0; i < row2; i++) {

        matrix2[i] = new int[col2];

    }

    //Input values

    cout<<"Enter value of matrix 1"<<endl;

    for(int i = 0; i < row1; i++) {

        for(int j = 0;j < col1; j++) {

            cout<<"["<< i + 1 << "]["<< j + 1 << "]=";

            cin>>matrix1[i][j];

        }

    }

    cout<<endl;

    cout<<"Enter value of matrix 2"<<endl;

    for(int i = 0; i < row2; i++) {

        for(int j = 0;j < col2; j++) {

            cout<<"["<< i + 1 << "]["<< j + 1 << "]=";

            cin>>matrix2[i][j];

        }

    }

    cout<<endl;

    //Matix Addition

    if(row1 == row2 && col1 == col2) {

        cout<<"Addition of two matrix"<<endl;

        for(int i = 0; i < row1; i++) {

            for(int j = 0;j < col1; j++) {

                cout<<matrix1[i][j] + matrix2[i][j] << " ";

            }

            cout<<endl;

        }

    } else {

        cout<<"Matrix Addition is not possible"<<endl;

    }

    cout<<endl;

    //Matix Subtraction

    if(row1 == row2 && col1 == col2) {

        cout<<"Subtraction of two matrix"<<endl;

        for(int i = 0; i < row1; i++) {

            for(int j = 0;j < col1; j++) {

                cout<<matrix1[i][j] - matrix2[i][j] << " ";

            }

            cout<<endl;

        }

    } else {

        cout<<"Matrix Subtraction is not possible"<<endl;

    }

    cout<<endl;

    //Matix Multiplication

    if(col1 == row2 ) {

        cout<<"Multiplication of two matrix"<<endl;

        int\*\* result = new int\*[row1];

        for (int i = 0; i < row1; i++) {

            result[i] = new int[col2]();

        }

        for(int i = 0; i < row1; i++) {

            for(int j = 0;j < col2; j++) {

                for(int k = 0; k < col1; k++) {

                    result[i][j] += matrix1[i][k] \* matrix2[k][j];

                }

            }

        }

        for(int i = 0; i < row1; i++) {

            for(int j = 0;j < col2; j++) {

                cout<<result[i][j]<<" ";

            }

            cout<<endl;

        }

        for (int i = 0; i < row1; i++) {

            delete[] result[i];

        }

        delete[] result;

    } else {

        cout<<"Matrix Multiplication is not possible"<<endl;

    }

    cout<<endl;

    //Free memory

    for (int i = 0; i < row1; i++) {

            delete[] matrix1[i];

    }

    delete[] matrix1;

    for (int i = 0; i < row2; i++) {

            delete[] matrix2[i];

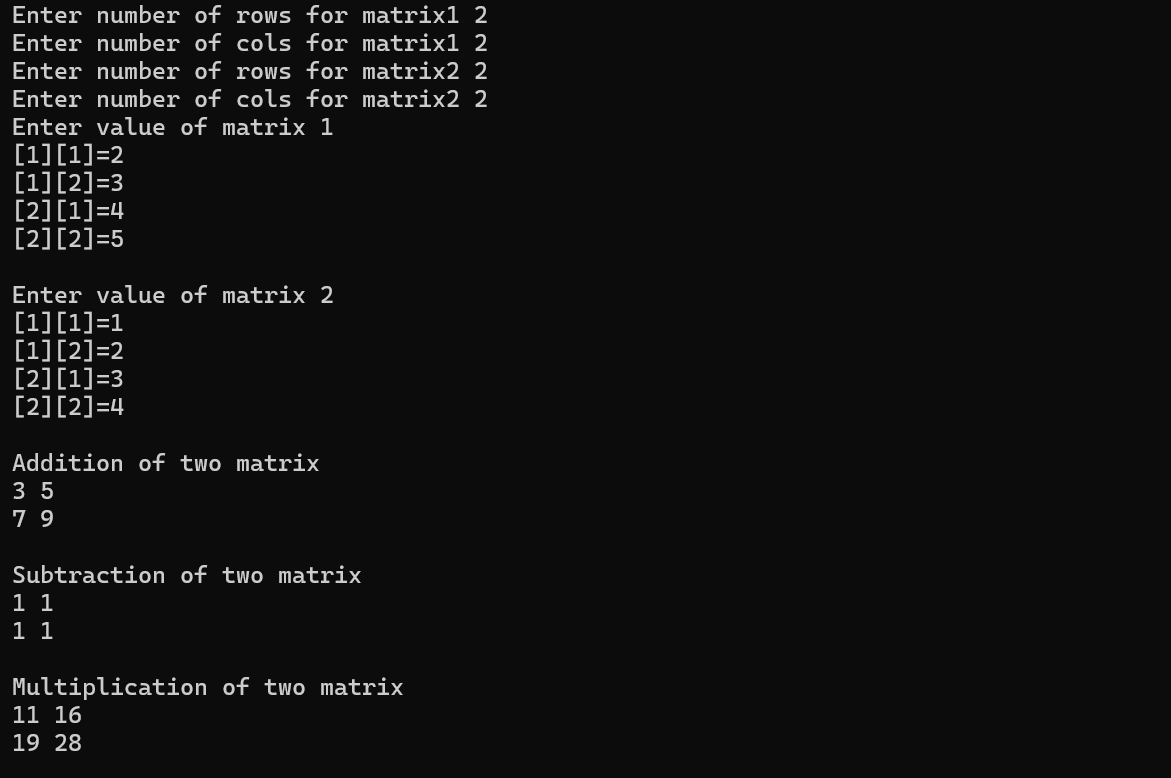
    }

    delete[] matrix2;

    return 0;

}

**Output**

****

**Task3:**

#include<iostream>

using namespace std;

struct employees {

    int employeeID;

    string name;

    string department;

    int salary;

} typedef employees;

void displayEmployees(employees\* emps, int size) {

    cout<<"\n---------------------------------------------------------" << endl << endl;

    for (int i = 0; i < size; i++) {

        cout << "Employee ID: " << emps[i].employeeID << endl;

        cout << "Employee Name: " << emps[i].name << endl;

        cout << "Department: " << emps[i].department << endl;

        cout << "Salary: " << emps[i].salary << endl;

        cout << endl;

        cout<<"\n---------------------------------------------------------" << endl << endl;

    }

}

void searchEmployee(employees\* emps, int size) {

    int id;

    cout << "Enter the employee ID to search: ";

    cin >> id;

    int i;

    for (i = 0; i < size; i++) {

        if (emps[i].employeeID == id) {

            cout << "\nEmployee found" << endl<<endl;

            break;

        }

    }

    if(i == size) cout << "\nEmployee not found" << endl<<endl;

}

int main() {

    int n;

    cout << "Enter total number of employees: ";

    cin >> n;

    employees\* emps = new employees[n];

    for(int i = 0; i < n; i++) {

        cout << "\nEnter employee ID: ";

        cin >> emps[i].employeeID;

        cout << "Enter employee name: ";

        cin >> emps[i].name;

        cout << "Enter employee department: ";

        cin >> emps[i].department;

        cout << "Enter employee salary: ";

        cin >> emps[i].salary;

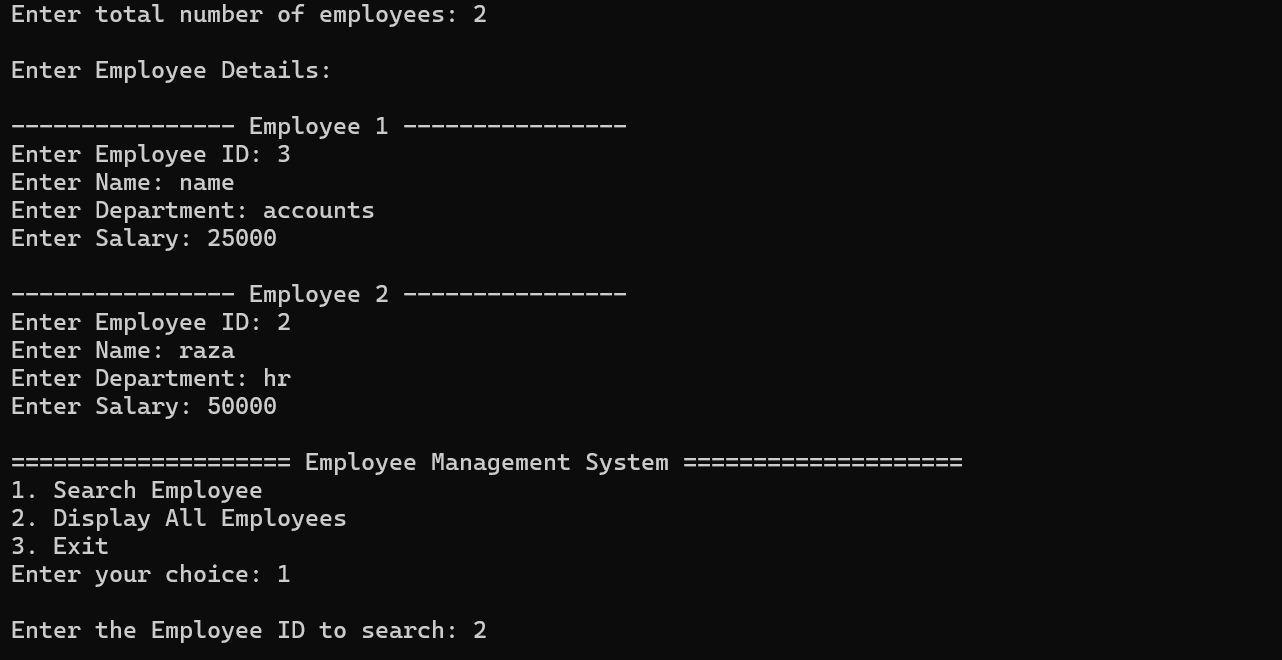
    }

    searchEmployee(emps,n);

    displayEmployees(emps,n);

}

**Output**

****

**A screenshot of a computer

Description automatically generated**

**Task4:**

#include <iostream>

#include <iomanip>

using namespace std;

struct Students {

    string name;

    int rollnumber;

    float \*marks = new float[5];

} typedef Students;

char grade(float avg) {

    if (avg >= 80.0) return 'A';

    if (avg >= 70.0) return 'B';

    if (avg >= 60.0) return 'C';

    if (avg >= 50.0) return 'D';

    return 'F';

}

void report(const Students& student, float avg, char grade) {

    cout << "\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

    cout << "              STUDENT REPORT CARD                \n";

    cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

    cout << " Name      : " << student.name << endl;

    cout << " Roll No.  : " << student.rollnumber << endl;

    cout << "-------------------------------------------------\n";

    cout << " Subject Marks: \n";

    for (int i = 0; i < 5; i++) {

        cout << "  - Subject " << i + 1 << " : " << student.marks[i] << endl;

    }

    cout << "-------------------------------------------------\n";

    cout << " Average Marks: " << avg << endl;

    cout << " Grade        : " << grade << endl;

    cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n";

}

int main() {

    int n;

    cout << "Enter number of students: ";

    cin >> n;

    if (n <= 0) {

        cout << "Invalid number of students!\n";

        return 0;

    }

    Students \*students = new Students[n];

    float \*avg = new float[n];

    char \*grades = new char[n];

    // Data Entry

    for (int i = 0; i < n; i++) {

        cout << "\n================== Entering Data for Student " << i + 1 << " ==================\n";

        cout << "Enter Name: ";

        cin.ignore();

        getline(cin, students[i].name);

        cout << "Enter Roll Number: ";

        cin >> students[i].rollnumber;

        for (int j = 0; j < 5; j++) {

            cout << "Enter Marks for Subject " << j + 1 << ": ";

            cin >> students[i].marks[j];

        }

    }

    // Calculating Average and Assigning Grades

    for (int i = 0; i < n; i++) {

        float sum = 0;

        for (int j = 0; j < 5; j++) {

            sum += students[i].marks[j];

        }

        avg[i] = sum / 5.0;

        grades[i] = grade(avg[i]);

    }

    // Printing Student Reports

    for (int i = 0; i < n; i++) {

        report(students[i], avg[i], grades[i]);

    }

    for (int i = 0; i < n; i++) {

        delete[] students[i].marks;

    }

    delete[] students;

    delete[] avg;

    delete[] grades;

    return 0;

}

**Output**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**Task5:**

#include <iostream>

#include <string.h>

using namespace std;

int main() {

    string \*str1 = new string;

    string \*str2 = new string;

    string \*str = new string;

    cout << "Enter string 1: ";

    getline(cin, \*str1);

    cout << "Enter string 2: ";

    getline(cin, \*str2);

    // String Concatenation

    \*str = \*str1 + \*str2;

    cout << "String concatenation: " << \*str << endl;

    // String Comparison

    if (\*str1 < \*str2) {

        cout << \*str1 << " is lexicographically smaller than " << \*str2 << endl;

    } else if (\*str1 > \*str2) {

        cout << \*str1 << " is lexicographically greater than " << \*str2 << endl;

    } else {

        cout << \*str1 << " is lexicographically equal to " << \*str2 << endl;

    }

    // String length

    cout << "String 1 length: " << str1->length() << endl;

    cout << "String 2 length: " << str2->length() << endl;

    cout << "Concatenated string length: " << str->length() << endl;

    delete str1;

    delete str2;

    delete str;

    return 0;

}

**Output**

**A screen shot of a computer

Description automatically generated**

**Task6:**

#include <iostream>

#include <string.h>

using namespace std;

int main() {

    string \*str1 = new string;

    string \*str2 = new string;

    string \*str = new string;

    cout << "Enter string 1: ";

    getline(cin, \*str1);

    cout << "Enter string 2: ";

    getline(cin, \*str2);

    // String Concatenation

    \*str = \*str1 + \*str2;

    cout << "String concatenation: " << \*str << endl;

    // String Comparison

    if (\*str1 < \*str2) {

        cout << \*str1 << " is lexicographically smaller than " << \*str2 << endl;

    } else if (\*str1 > \*str2) {

        cout << \*str1 << " is lexicographically greater than " << \*str2 << endl;

    } else {

        cout << \*str1 << " is lexicographically equal to " << \*str2 << endl;

    }

    // String length

    cout << "String 1 length: " << str1->length() << endl;

    cout << "String 2 length: " << str2->length() << endl;

    cout << "Concatenated string length: " << str->length() << endl;

    delete str1;

    delete str2;

    delete str;

    return 0;

}

**Output**

**A screenshot of a computer program

Description automatically generated**

**A screen shot of a computer

Description automatically generated**

**Task7:**

#include <iostream>

using namespace std;

// Function to swap two integers using pointers

void swap(int \*a, int \*b) {

    int temp = \*b;

    \*b = \*a;

    \*a = temp;

}

// Function to reverse an array using pointers

void reverseArray(int \*array, int size) {

    int i = 0, j = size - 1;

    while (i < j) {

        swap(&array[i++], &array[j--]);

    }

}

int main() {

    int n;

    cout << "Enter size of array: ";

    cin >> n;

    int \*array = new int[n];

    for (int i = 0; i < n; i++) {

        cout << "Enter number " << i + 1 << " : ";

        cin >> array[i];

    }

    cout << "\nEntered array:  [ ";

    for (int i = 0; i < n; i++) {

        cout << array[i] << " ";

    }

    cout << "]\n";

    // Reverse array

    reverseArray(array, n);

    cout << "Reversed array: [ ";

    for (int i = 0; i < n; i++) {

        cout << array[i] << " ";

    }

    cout << "]\n";

    delete[] array;

    return 0;

}

**Output**

**A black screen with white text

Description automatically generated**

**Task8:**

#include <iostream>

#include <string>

using namespace std;

struct Product {

    int productID;

    string name;

    int quantity;

    double price;

};

// Function to add a new product

void addProduct(Product \*&inventory, int &size) {

    Product \*newInventory = new Product[size + 1];

    for (int i = 0; i < size; i++) {

        newInventory[i] = inventory[i];

    }

    cout << "Enter Product ID: ";

    cin >> newInventory[size].productID;

    cout << "Enter Product Name: ";

    cin.ignore();

    getline(cin, newInventory[size].name);

    cout << "Enter Quantity: ";

    cin >> newInventory[size].quantity;

    cout << "Enter Price: ";

    cin >> newInventory[size].price;

    delete[] inventory;

    inventory = newInventory;

    size++;

}

// Function to remove a product

void removeProduct(Product \*&inventory, int &size, int productID) {

    int index = -1;

    for (int i = 0; i < size; i++) {

        if (inventory[i].productID == productID) {

            index = i;

            break;

        }

    }

    if (index == -1) {

        cout << "Product not found!" << endl;

        return;

    }

    Product \*newInventory = new Product[size - 1];

    for (int i = 0, j = 0; i < size; i++) {

        if (i != index) {

            newInventory[j++] = inventory[i];

        }

    }

    delete[] inventory;

    inventory = newInventory;

    size--;

    cout << "Product removed successfully!" << endl;

}

// Function to update a product

void updateProduct(Product \*inventory, int size, int productID) {

    for (int i = 0; i < size; i++) {

        if (inventory[i].productID == productID) {

            cout << "Enter new name: ";

            cin.ignore();

            getline(cin, inventory[i].name);

            cout << "Enter new quantity: ";

            cin >> inventory[i].quantity;

            cout << "Enter new price: ";

            cin >> inventory[i].price;

            cout << "Product updated successfully!" << endl;

            return;

        }

    }

    cout << "Product not found!" << endl;

}

// Function to display inventory

void displayInventory(Product \*inventory, int size) {

    if (size == 0) {

        cout << "Inventory is empty!" << endl;

        return;

    }

    cout << "\nInventory List:" << endl;

    cout << "------------------------------------------" << endl;

    cout << "ID\tName\t Quantity\tPrice" << endl;

    cout << "------------------------------------------" << endl;

    for (int i = 0; i < size; i++) {

        cout << inventory[i].productID << "\t" << inventory[i].name << "\t\t"

             << inventory[i].quantity << "\t\t$" << inventory[i].price << "" << endl;

    }

    cout << "------------------------------------------" << endl;

}

// Function to calculate total inventory value

double calculateTotalValue(Product \*inventory, int size) {

    double totalValue = 0;

    for (int i = 0; i < size; i++) {

        totalValue += inventory[i].price \* inventory[i].quantity;

    }

    return totalValue;

}

int main() {

    Product \*inventory = nullptr;

    int size = 0;

    int choice, productID;

    do {

        cout << "\nInventory Management System" << endl;

        cout << "1. Add Product\n2. Remove Product\n3. Update Product" << endl;

        cout << "4. Display Inventory\n5. Calculate Total Value\n6. Exit" << endl;

        cout << "Enter your choice: ";

        cin >> choice;

        switch (choice) {

            case 1:

                addProduct(inventory, size);

                break;

            case 2:

                cout << "Enter Product ID to remove: ";

                cin >> productID;

                removeProduct(inventory, size, productID);

                break;

            case 3:

                cout << "Enter Product ID to update: ";

                cin >> productID;

                updateProduct(inventory, size, productID);

                break;

            case 4:

                displayInventory(inventory, size);

                break;

            case 5:

                cout << "Total Inventory Value: $" << calculateTotalValue(inventory, size) << endl;

                break;

            case 6:

                cout << "Exiting program..." << endl;

                break;

            default:

                cout << "Invalid choice! Please try again." << endl;

        }

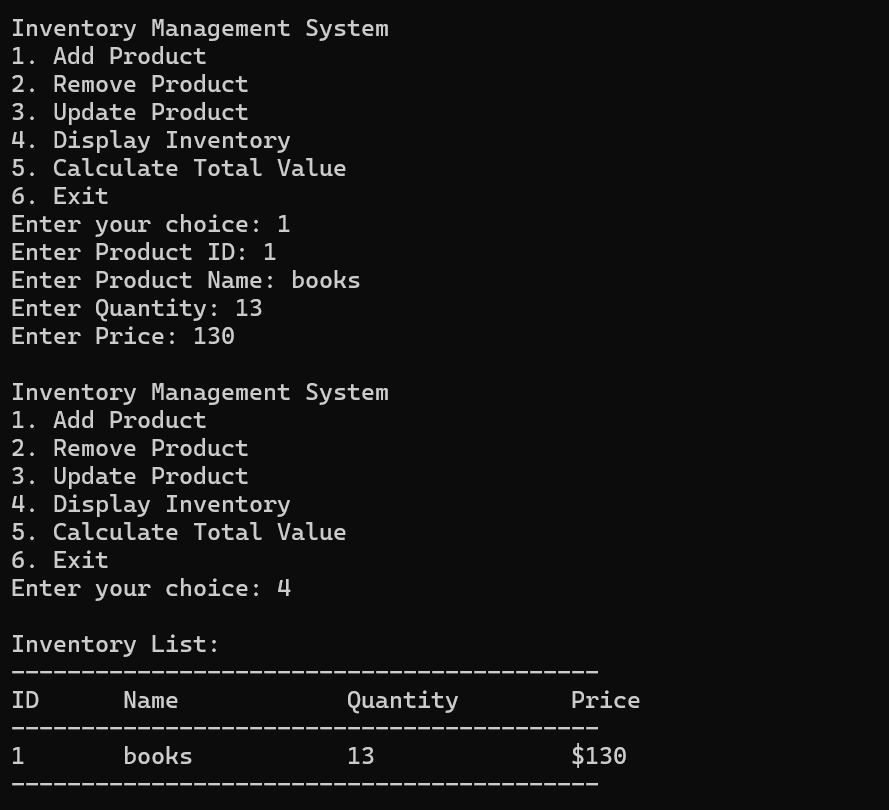
    } while (choice != 6);

    delete[] inventory;

    return 0;

}

**Output**

****

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer screen

Description automatically generated**

**Task9:**

#include <iostream>

using namespace std;

struct Student {

    string name;

    int rollNumber;

    float \*marks = new float[5];

} typedef Student;

Student findTopper(Student\* students, int n) {

    int topIndex = 0;

    float highestMarks = 0, sum = 0;

    for (int i = 0; i < n; i++) {

        sum = 0;

        for (int j = 0; j < 5; j++) {

            sum += students[i].marks[j];

        }

        if (sum > highestMarks) {

            highestMarks = sum;

            topIndex = i;

        }

    }

    return students[topIndex];

}

int main() {

    int n;

    cout << "Enter number of students: ";

    cin >> n;

    Student\* students = new Student[n];

    // Input student data

    for (int i = 0; i < n; i++) {

        cout << "\nEnter Student " << i + 1 << " Name: ";

        cin >> students[i].name;

        cout << "Enter Roll Number: ";

        cin >> students[i].rollNumber;

        cout << "Enter 5 subject marks: ";

        for(int j = 0; j < 5; j++) {

            cout<<"Enter student " << i + 1 << " Subject Marks " << j + 1<<" : ";

            cin >> students[i].marks[j];

        }

    }

    // Highest marks student

    Student topper = findTopper(students, n);

    cout << "\n\*\*\*\*\*\* Student with Highest Marks \*\*\*\*\*\*\n";

    cout << "Name: " << topper.name << endl;

    cout << "Roll Number: " << topper.rollNumber << endl;

    cout << "Marks: ";

    for (int j = 0; j < 5; j++) {

        cout << topper.marks[j] << " ";

    }

    cout << endl;

    for (int i = 0; i < n; i++) {

        delete[] students[i].marks;

    }

    delete[] students;

    return 0;

}

**Output**

**A screenshot of a computer

Description automatically generated**

**Task10:**

#include <iostream>

using namespace std;

int main() {

    int rows, cols;

    cout << "Enter number of rows: ";

    cin >> rows;

    cout << "Enter number of columns: ";

    cin >> cols;

    int\*\* matrix = new int\*[rows];

    for (int i = 0; i < rows; i++) {

        matrix[i] = new int[cols];

    }

    // Input Matrix Elements

    cout << "Enter the elements of the matrix:" << endl;

    for (int i = 0; i < rows; i++) {

        for (int j = 0; j < cols; j++) {

            cout << "Element [" << i + 1 << "][" << j + 1 << "]: ";

            cin>>matrix[i][j];

        }

    }

    int\*\* transpose = new int\*[cols];

    for (int i = 0; i < cols; i++) {

        transpose[i] = new int[rows];

    }

    //calculating transpose

    for (int i = 0; i < rows; i++) {

        for (int j = 0; j < cols; j++) {

            \*(\*(transpose + j) + i) = \*(\*(matrix + i) + j);

        }

    }

    // Display Original Matrix

    cout << "\nOriginal Matrix:" << endl;

    for (int i = 0; i < rows; i++) {

        for (int j = 0; j < cols; j++) {

            cout << \*(\*(matrix + i) + j) << " ";

        }

        cout << endl;

    }

    // Display Transposed Matrix

    cout << "\nTransposed Matrix:" << endl;

    for (int i = 0; i < cols; i++) {

        for (int j = 0; j < rows; j++) {

            cout << \*(\*(transpose + i) + j) << " ";

        }

        cout << endl;

    }

    for (int i = 0; i < rows; i++) {

        delete[] matrix[i];

    }

    delete[] matrix;

    for (int i = 0; i < cols; i++) {

        delete[] transpose[i];

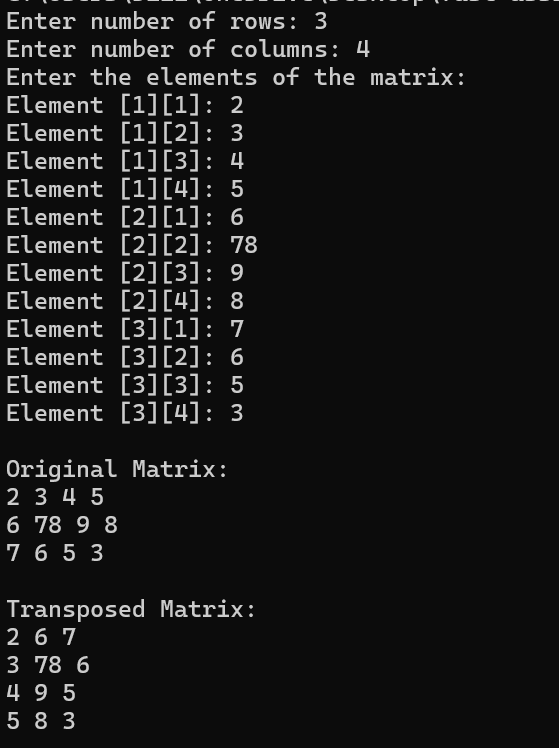
    }

    delete[] transpose;

    return 0;

}

**Output**

****