Lab 02:

Q1:

#include <iostream>

using namespace std;

int main()

{

    int n, index;

    float sum = 0.0, average;

    cout << "Enter the number of elements in array you want: ";

    cin >> n;

    float \*ptr = new float[n];

    for (int i = 0; i < n; i++)

    {

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

        cin >> \*(ptr + i);

        sum += ptr[i];

    }

    average = sum / n;

    float a1, diff = 100.0;

    for (int i = 0; i < n; i++)

    {

        a1 = average - ptr[i];

        if (a1 < 0)

            a1 = a1 \* (-1);

        if (a1 < diff)

        {

            diff = a1;

            index = i;

        }

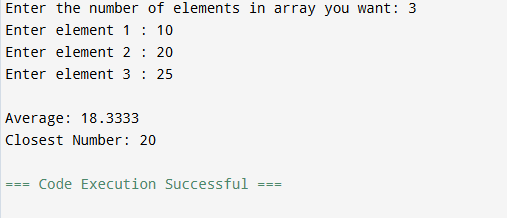
    }

    cout << "\nAverage: " << average << endl;

    cout << "Closest Number: " << ptr[index];

}

Output:



Q2:

#include <iostream>

#include <string>

using namespace std;

int main()

{

    int n;

    cout << "Enter the size of string array you want: ";

    cin >> n;

    cin.ignore();

    string \*ptr = new string[n];

    int \*ptr2 = new int[n];

    for (int i = 0; i < n; i++)

    {

        cout << "String " << i + 1 << " : ";

        getline(cin, \*(ptr + i));

    }

    for (int i = 0; i < n; i++)

    {

        int m = ptr[i].length();

        cout << "\nReverse string " << i + 1 << " : ";

        for (int j = m; j >= 0; j--)

        {

            cout << ptr[i][j];

        }

    }

    for (int i = 0; i < n; i++)

    {

        int counter = 0;

        int n = ptr[i].length();

        for (int j = 0; j < n; j++)

        {

            if (ptr[i][j] == 'a' || ptr[i][j] == 'e' || ptr[i][j] == 'i' || ptr[i][j] == 'o' || ptr[i][j] == 'u')

                counter++;

            if (ptr[i][j] == 'A' || ptr[i][j] == 'E' || ptr[i][j] == 'I' || ptr[i][j] == 'O' || ptr[i][j] == 'U')

                counter++;

        }

        ptr2[i] = counter;

    }

    int max = 0, index;

    for (int i = 0; i < n; i++)

    {

        if (ptr2[i] > max)

        {

            max = ptr2[i];

            index = i;

        }

    }

    cout << "\nString that contain most vowels is" << endl;

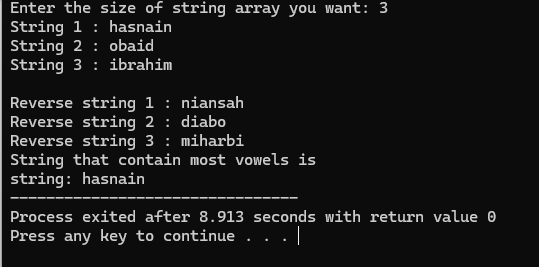
    cout << "string: " << ptr[index];

    delete []ptr;

    delete []ptr2;

}

Output:



Q3:

#include <iostream>

using namespace std;

int \*\*Allocate\_memory(int row, int colom);

void initializearray(int \*\*arr, int row, int colom);

void sum(int \*\*arr, int row, int colom);

void sum\_each\_row\_colom(int \*\*arr, int row, int colom);

void highest\_row\_colom(int \*\*arr, int row, int colom);

void memoryfree(int \*\*arr, int row);

int main()

{

    int row, colom;

    srand(time(0));

    cout << "Enter the row: ";

    cin >> row;

    cout << "Enter the coloum: ";

    cin >> colom;

    int \*\*arr = Allocate\_memory(row, colom);

    if (!arr)

    {

        cout << "Error in memory allocation" << endl;

        return 1;

    }

    initializearray(arr, row, colom);

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

    for (int i = 0; i < row; i++)

    {

        cout << "[";

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

        {

            cout << arr[i][j] << ",";

        }

        cout << "]\n";

    }

    sum(arr, row, colom);

    sum\_each\_row\_colom(arr, row, colom);

    highest\_row\_colom(arr, row, colom);

    cout << "Transpose Matrix" << endl;

    for (int i = 0; i < colom; i++)

    {

        cout << "[";

        for (int j = 0; j < row; j++)

        {

            cout << arr[j][i] << ",";

        }

        cout << "]\n";

    }

    memoryfree(arr, row);

}

int \*\*Allocate\_memory(int row, int colom)

{

    int \*\*arr = new int \*[row];

    if (!arr)

    {

        cout << "Error in memory allocation" << endl;

        return nullptr;

    }

    for (int i = 0; i < row; i++)

    {

        arr[i] = new int[colom];

        if (!arr[i])

        {

            cout << "Error in memory allocation" << endl;

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

                delete[] arr[j];

            delete[] arr;

            return nullptr;

        }

    }

    return arr;

}

void initializearray(int \*\*arr, int row, int colom)

{

    for (int i = 0; i < row; i++)

    {

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

        {

            arr[i][j] = rand() % 100 + 1;

        }

    }

}

void sum(int \*\*arr, int row, int colom)

{

    int sum = 0;

    for (int i = 0; i < row; i++)

    {

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

        {

            sum += arr[i][j];

        }

    }

    cout << "Sum: " << sum << endl;

}

void sum\_each\_row\_colom(int \*\*arr, int row, int colom)

{

    cout << "Row Sums: ";

    for (int i = 0; i < row; i++)

    {

        int sum = 0;

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

        {

            sum += arr[i][j];

        }

        cout << sum << ",";

    }

    cout << "\nColumn Sums: ";

    for (int i = 0; i < colom; i++)

    {

        int sum = 0;

        for (int j = 0; j < row; j++)

        {

            sum += arr[j][i];

        }

        cout << sum << ",";

    }

}

void highest\_row\_colom(int \*\*arr, int row, int colom)

{

    int \*rowsum = new int[row];

    int max = 0, index;

    for (int i = 0; i < row; i++)

    {

        int sum = 0;

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

        {

            sum += arr[i][j];

        }

        rowsum[i] = sum;

    }

    for (int i = 0; i < row; i++)

    {

        if (rowsum[i] > max)

        {

            max = rowsum[i];

            index = i;

        }

    }

    cout << "\nRow with highest Sum: Row " << index + 1 << endl;

    int \*colsum = new int[colom];

    int max2 = 0, index2;

    for (int i = 0; i < colom; i++)

    {

        int sum = 0;

        for (int j = 0; j < row; j++)

        {

            sum += arr[j][i];

        }

        colsum[i] = sum;

    }

    for (int i = 0; i < colom; i++)

    {

        if (colsum[i] > max2)

        {

            max2 = colsum[i];

            index2 = i;

        }

    }

    cout << "Column with highest Sum: Column " << index2 + 1 << endl;

    delete[] rowsum;

    delete[] colsum;

}

void memoryfree(int \*\*arr, int row)

{

    for (int i = 0; i < row; i++)

    {

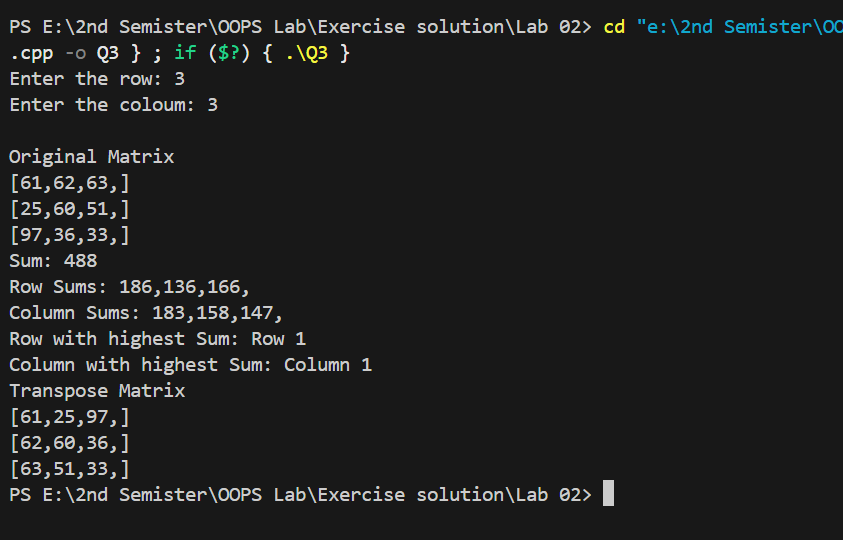
        delete[] arr[i];

    }

    delete[] arr;

}

Output:



Q4:

#include<iostream>

using namespace std;

int\* memory\_allocation(int size);

void uniquie(int \*arr,int size,int \*result,int \*numsize);

int main(){

int\* array,\*result;

int numsize,n;

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

cin>>n;

array=memory\_allocation(n);

result=memory\_allocation(n);

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

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

    cin>>array[i];

}

uniquie(array,n,result,&numsize);

cout<<"\nUnique Elements are: ";

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

    cout<<result[i]<<",";

}

}

int\* memory\_allocation(int size){

    int \*array = new int[size];

    if (!array)

    {

        cout << "Error in memory allocation" << endl;

        return nullptr;

    }

   return array;

}

void uniquie(int \*arr,int size,int \*result,int \*numsize){

   \*numsize=0;

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

     int found=0;

     for(int j=0;j<\*numsize;j++){

        if(arr[i]==result[j]){

          found=1;

          break;}

     }

        if(!found){

            result[\*numsize]=arr[i];

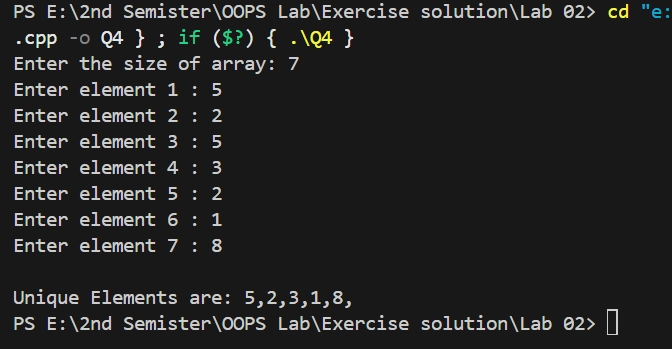
            (\*numsize)++;

        }

}

}

Output:



Q5:

#include<iostream>

#include<cstring>

using namespace std;

void swaping(char\* str,int shift,int times);

int main(){

char str[100];

int n;

cout<<"enter String: ";

 cin.getline(str, 100);

cout<<"Enter the numbers of characters to shift: ";

cin>>n;

swaping(str,n,n);

cout<<"Shifted string after shifting last "<<n<<" characters: "<<str<<endl;

}

void swaping(char\* str,int shift,int times){

    int len=strlen(str);

    char temp[len];

    for(int i=0;i<times;i++){

        strncpy(temp,str+len-shift,shift);

        strncpy(temp+shift,str,len-shift);

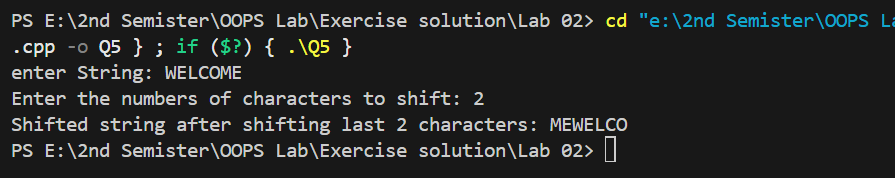
        temp[len]='\0';

    }

     strcpy(str,temp);

}

Output:



Q6:

#include <iostream>

#include <string>

using namespace std;

struct Register

{

    int course\_id;

    string course\_name;

};

struct Student

{

    struct Register s2;

    int student\_id;

    double cellno;

    string firstname, lastname, email;

};

int main()

{

    struct Student s1[5];

    cout << "-----Enter the Students Details-----" << endl;

    for (int i = 0; i < 5; i++)

    {

        cout << " Student " << i + 1 << " " << endl;

        cout << "Course id: ";

        cin >> s1[i].s2.course\_id;

        cin.ignore();

        cout << "Course name: ";

        getline(cin, s1[i].s2.course\_name);

        cout << "Student id: ";

        cin >> s1[i].student\_id;

        cin.ignore();

        cout << "First name: ";

        getline(cin, s1[i].firstname);

        cout << "Last name: ";

        getline(cin, s1[i].lastname);

        cout << "Cell no: ";

        cin >> s1[i].cellno;

        cin.ignore();

        cout << "Email: ";

        getline(cin, s1[i].email);

    }

    for (int i = 0; i < 5; i++)

    {

        cout << "----------Student " << i + 1 << " ---------" << endl;

        cout << "Course id: " << s1[i].s2.course\_id << endl;

        cout << "Course Name: " << s1[i].s2.course\_name << endl;

        cout << "Student id: " << s1[i].student\_id << endl;

        cout << "Name: " << s1[i].firstname << " " << s1[i].lastname << endl;

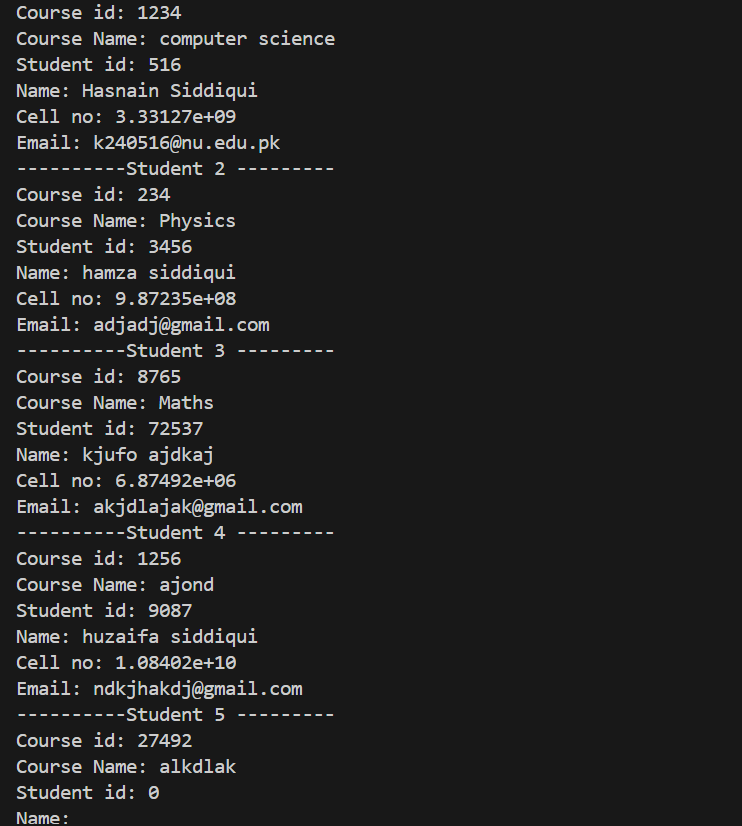
        cout << "Cell no: " << s1[i].cellno << endl;

        cout << "Email: " << s1[i].email << endl;

    }

}

Output:



Q7:

#include <iostream>

#include <cstdlib>

using namespace std;

struct Product {

    int product\_id;

    string name;

    int quantity;

    float price;

};

Product\* products = nullptr;

int product\_count = 0;

int capacity = 2;

void addproduct(int id, string n1, float p1, int q1);

void product\_finder(int pro\_id);

void update\_product(int pro\_id, float p1, int q1);

void remove\_product(int pro\_id);

int main() {

    products = (Product\*)malloc(capacity \* sizeof(Product));

    addproduct(1, "Laptop", 799.99, 50);

    addproduct(2, "Smartphone", 499.99, 100);

    product\_finder(1);

    update\_product(1, 749.99, 45);

    product\_finder(1);

    remove\_product(2);

    product\_finder(2);

    free(products);

    return 0;

}

void addproduct(int id, string n1, float p1, int q1) {

    if (product\_count >= capacity) {

        capacity \*= 2;

        products = (Product\*)realloc(products, capacity \* sizeof(Product));

        if (!products) {

            cout << "Memory allocation failed" << endl;

            exit(1);

        }

    }

    products[product\_count].product\_id = id;

    products[product\_count].name = n1;

    products[product\_count].quantity = q1;

    products[product\_count].price = p1;

    product\_count++;

    cout << "Product added successfully" << endl;

}

void product\_finder(int pro\_id) {

    for (int i = 0; i < product\_count; i++) {

        if (products[i].product\_id == pro\_id) {

            cout << "Product id: " << products[i].product\_id << endl;

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

            cout << "Price: " << products[i].price << endl;

            cout << "Quantity in Stock: " << products[i].quantity << endl;

            return;

        }

    }

    cout << "Sorry, product not found" << endl;

}

void update\_product(int pro\_id, float p1, int q1) {

    for (int i = 0; i < product\_count; i++) {

        if (products[i].product\_id == pro\_id) {

            products[i].price = p1;

            products[i].quantity = q1;

            cout << "Record updated successfully" << endl;

            return;

        }

    }

    cout << "Sorry, product not found" << endl;

}

void remove\_product(int pro\_id) {

    for (int i = 0; i < product\_count; i++) {

        if (products[i].product\_id == pro\_id) {

            for (int j = i; j < product\_count - 1; j++) {

                products[j] = products[j + 1];

            }

            product\_count--;

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

            return;

        }

    }

    cout << "Sorry, product not found" << endl;

}

Output:

