1/27/2025

Muhammad Arham Usman (24k-1016)

FAST NUCES

Object Oriented Programming

Lab 2

QUESTION 01

# SOURCE CODE:

/\*

Write a C++ program that reads a group of n numbers from the user and stores them in a dynamically

allocated array of type float. Then, the program should:

- Calculate the average of the numbers.

- Find the number closest to the average.

- Print the average and the number closest to it.

- Use pointer notation wherever possible.

\*/

#include <iostream>

using namespace std;

float number(float \*arr, float avg, int size){

    float num, diff=999999.9;

    for (int i=0; i<size; i++){

        float diff2=arr[i]-avg;

        if (diff2<0) diff2\*=-1;

        if (diff2<diff) {

            num=arr[i]; diff=diff2;

        }

    }

    return num;

}

int main(){

    int size=0;

    cout<<"Enter number of values that you want to input: ";

    cin>>size;

    float \*userInput=new float[size], sum=0;

    for (int i=0; i<size; i++){

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

        cin>>userInput[i];

        sum+=userInput[i];

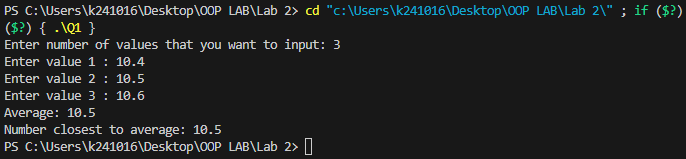
    }

    float average=sum/size, num=number(userInput, average, size);

    cout<<"Average: "<<average<<"\nNumber closest to average: "<<num;

}

# OUTPUT:



QUESTION 2

# SOURCE CODE:

/\*

Write a C++ program that:

- Reads n strings from the user and stores them in a dynamically allocated array of char\*.

- Prints the strings in reverse order using pointer arithmetic.

- Finds and prints the string with the most vowels (a, e, i, o, u).

- Calculates and prints the average length of all the strings.

Note: Use pointer notation wherever possible.

If there are multiple strings with the same number of vowels, print the first one encountered

\*/

#include <iostream>

#include <string>

using namespace std;

void print\_str\_in\_rev(char \*c, int size){

    for (int i=size-1; i>=0; i--){

        cout<<c[i];

    }

    cout<<endl;

}

int vowels(char \*c){

    int v=0;

    for (int i=0; c[i]!='\0'; i++){

        if (c[i]=='A'||c[i]=='E'||c[i]=='I'||c[i]=='O'||c[i]=='U'||c[i]=='a'||c[i]=='e'||c[i]=='i'||c[i]=='o'||c[i]=='u'){

            v++;

        }

    }

    return v;

}

int max\_vowels(char\*\*c, int size){

    int v=-1; int index=-1; int num;

    for (int i=0; i<size; i++){

        num=vowels(c[i]);

        if (num>v){

            index=i;

            v=num;

        }

    }

    return index;

}

float average(int \*arr, int size){

    float avg=0;

    for (int i=0; i<size; i++){

        avg+=arr[i];

    }

    return avg/size;

}

int main(){

    int size;

    cout<<"Enter number of strings that you want to input: ";

    cin>>size;

    char \*\*sptr= new char\*[size];

    int \*sizes=new int[size];

    string gg;

    for (int i=0; i<size; i++){

        cout<<"Enter string: ";

        cin>>gg;

        sizes[i]=gg.length()+1;

        sptr[i]=new char[sizes[i]];

        for (int j=0; j<sizes[i];j++){

            sptr[i][j]=gg[j];

        }

    }

    cout<<"Print strings in reverse\n";

    for (int i=0; i<size; i++){

        print\_str\_in\_rev(sptr[i], sizes[i]-1);

    }

    cout<<"\nString with most vowels: ";

    int max=max\_vowels(sptr, size);

    for (int i=0; sptr[max][i]!='\0'; i++){

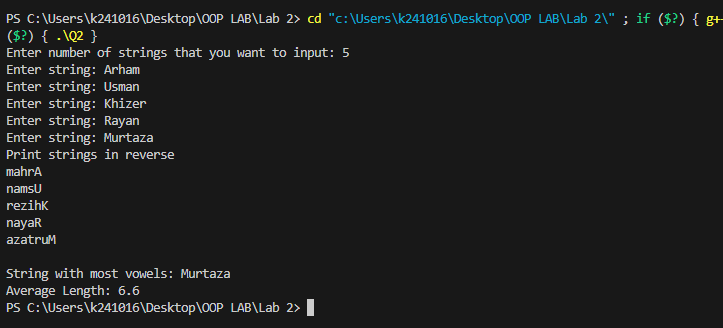
        cout<<sptr[max][i];

    }

    cout<<"\nAverage Length: "<<average(sizes, size);

}

# OUTPUT:



QUESTION 3

# SOURCE CODE:

/\*

Write a C++ program that:

- Dynamically allocates a 2D array using pointers (not using vector or standard containers).

- Fills the array with random integers between 1 and 100.

- Pass the 2D array to function to perform these tasks:

- Calculates and prints The sum of all elements in the array.

- Calculates and prints The sum of each row and each column.

- Calculates and prints The row and column with the highest sum.

- Pass the 2D array to a function to transpose the matrix and print the resulting matrix.

Free the dynamically allocated memory.

Note:

Use functions to perform the calculations and matrix operations (do not write all code inside main()).

Handle edge cases, such as when the array has no elements or is improperly allocated.

Sample Output:

Original Matrix:

[ 12 35 56 ]

[ 8 45 67 ]

[ 23 54 34 ]

Sum of all elements: 434

Row sums: 103, 120, 111

Column sums: 43, 134, 157

Row with highest sum: Row 2

Column with highest sum: Column 3

Transposed Matrix:

[ 12 8 23 ]

[ 35 45 54 ]

[ 56 67 34 ]

\*/

#include <iostream>

#include <math.h>

#include <cstdlib>

using namespace std;

void total\_sum(int \*\*&arr){

    cout<<"Sum of all elements: ";

    int s=0;

    for (int i=0; i<3; i++){

        for (int j=0; j<3; j++){

            s+=arr[i][j];

        }

    }

    cout<<s<<endl;

}

int sum\_row(int \*\*&arr){

    cout<<"Row sums: ";

    int s=0, index=-1, max\_sum=0;

    for (int i=0; i<3; i++){

        s=0;

        for (int j=0; j<3; j++){

            s+=arr[i][j];

        }

        if (i!=2) cout<<s<<", ";

        else cout<<s<<endl;

        if (max\_sum<s){

            max\_sum=s;

            index=i;

        }

    }

    return index;

}

int sum\_col(int \*\*&arr){

    cout<<"Column sums: ";

    int s=0, index=-1, max\_sum=0;

    for (int i=0; i<3; i++){

        s=0;

        for (int j=0; j<3; j++){

            s+=arr[j][i];

        }

        if (i!=2) cout<<s<<", ";

        else cout<<s<<endl;

        if (max\_sum<s){

            max\_sum=s;

            index=i;

        }

    }

    return index;

}

int\*\* transpose(int \*\*&arr){

    int \*\*t= new int\*[3];

    if (t==NULL){

        cout<<"Memory allocation Failed:(";

        return t;

    }

    for (int i=0; i<3; i++){

        t[i]=new int[3];

        if (t[i]==NULL){

            cout<<"Memory allocation Failed:(";

            return t;

        }

    }

    for (int i=0; i<3; i++){

        for (int j=0; j<3; j++){

            t[i][j]=arr[j][i];

        }

    }

    return t;

}

void print\_arr(int \*\*&arr){

    for (int i=0; i<3; i++){

        for (int j=0; j<3; j++){

            cout<<arr[i][j]<<" ";

        }

        cout<<endl;

    }

}

void master\_function(int \*\*&arr){

    cout<<"Original Matrix:\n";

    print\_arr(arr);

    cout<<endl;

    total\_sum(arr);

    int row=sum\_row(arr);

    int col=sum\_col(arr);

    cout<<"Row with highest sum: Row "<<row+1<<endl;

    cout<<"Column with highest sum: Column "<<col+1<<endl;

    cout<<endl;

    int \*\*transposed\_matrix=transpose(arr);

    cout<<"Transposed Matrix:\n";

    print\_arr(transposed\_matrix);

    delete []transposed\_matrix;

}

int main(){

    int \*\*arr= new int\*[3];

    if (arr==NULL){

        cout<<"Memory allocation Failed:(";

        return -1;

    }

    for (int i=0; i<3; i++){

        arr[i]=new int[3];

        if (arr[i]==NULL){

            cout<<"Memory allocation Failed:(";

            return -1;

        }

        for (int j=0; j<3; j++){

            arr[i][j]=rand()%101;

            while (arr[i][j]<1) arr[i][j]=rand()%101;

        }

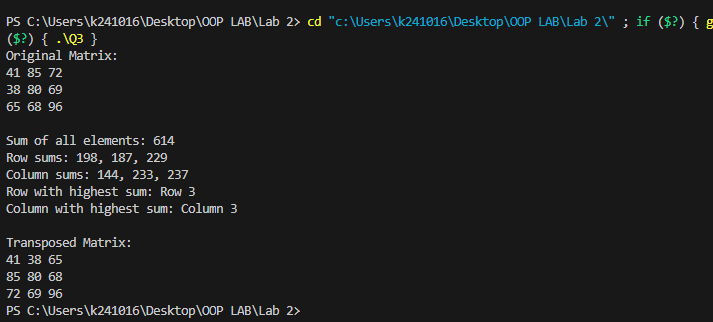
    }

    master\_function(arr);

    delete []arr;

}

# OUTPUT:



QUESTION 4

# SOURCE CODE:

/\*

You are required to write a C++ program that will creates a function named unique that will take array

as input . the array may contains the duplicates values but you have to process on the array and have to

return the array which must contains only unique values not duplicates.

\*/

#include <iostream>

using namespace std;

int present(int \*arr, int size, int value){

    for (int i=0; i<size; i++){

        if (arr[i]==value) return 1;

    }

    return 0;

}

int\* unique(int \*arr, int &size){

    int\* revised=new int[size], new\_size=0;

    for (int i=0; i<size; i++){

        if (present(revised, new\_size, arr[i])==0){

            revised[new\_size++]=arr[i];

        }

    }

    size=new\_size;

    return revised;

}

int main(){

    int size;

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

    cin>>size;

    int \*arr= new int[size];

    for (int i=0; i<size; i++){

        cin>>arr[i];

    }

    arr=unique(arr, size);

    cout<<"Unique elements in the array are: ";

    for (int i=0; i<size; i++){

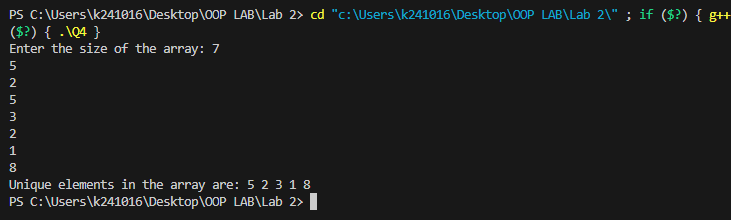
        cout << arr[i]<< " ";

    }

    delete[]arr;

}

# OUTPUT:



QUESTION 5

# SOURCE CODE:

/\*

You are required to write a c++ function swap\_string that shifts the last n characters of a string to the

front n times. It will take str and int as parameters. And will return the new string after shifting.

Note: You have to work with pointers.

\*/

#include <iostream>

using namespace std;

string swap\_string(string &s, int num){

    int len=s.length();

    string c;

    int index=len-num;

    if (num>len) num=len;

    for (int i=0; i<num; i++){

        c+=s[index++];

    }

    for (int i=num; i<len; i++){

        c+=s[i-num];

    }

    return c;

}

int main(){

    string s;

    cout<<"Enter a string: ";

    cin>>s;

    int num;

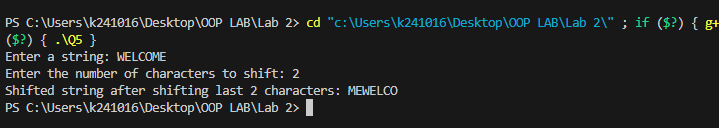
    cout<<"Enter the number of characters to shift: ";

    cin>>num;

    cout<<"Shifted string after shifting last "<<num<<" characters: "<<swap\_string(s, num);

}

# OUTPUT:



QUESTION 6

# SOURCE CODE:

/\*

You are tasked with implementing a simple Student Registration System in C++. Define two structures,

Register and Student, where Register contains attributes courseId and courseName, and Student inherits

from Register while having additional attributes such as studentId, firstName, lastName, cellNo, and

email. Your goal is to create an array of Student structures to store information for five students. Write a

C++ program that accomplishes the following tasks:

- Implement the Register and Student structures.

- Inherit the Register structure in the Student structure.

- Create an array of Student structures to store information for 5 students.

- Take input for each student, including their courseId, courseName, studentId, firstName,

lastName, cellNo, and email.

- Display the information for all 5 students.

\*/

#include <iostream>

using namespace std;

typedef struct{

    string courseID;

    string courseName;

}Register;

typedef struct{

    Register rgt;

    string studentId;

    string firstName;

    string LastName;

    string cellNo;

    string email;

}Student;

void input(Student &students){

    cout<<"Enter Course ID: ";

    fflush(stdout);

    cin>> students.rgt.courseID;

    cout<<"Enter Course Name: ";

    fflush(stdout);

    cin>> students.rgt.courseName;

    cout<<"Enter Student ID: ";

    fflush(stdout);

    cin>> students.studentId;

    cout<<"Enter First Name: ";

    fflush(stdout);

    cin>> students.firstName;

    cout<<"Enter Last Name: ";

    fflush(stdout);

    cin>> students.LastName;

    cout<<"Enter cell no: ";

    fflush(stdout);

    cin>> students.cellNo;

    cout<<"Enter email: ";

    fflush(stdout);

    cin>> students.email;

}

void display(Student &students){

    cout<<"Course ID: "<<students.rgt.courseID;

    cout<<"\nCourse Name: "<<students.rgt.courseName;

    cout<<"\nStudent ID: "<<students.studentId;

    cout<<"\nFirst Name: "<<students.firstName;

    cout<<"\nLast Name: "<<students.LastName;

    cout<<"\ncell no: "<<students.cellNo;

    cout<<"\nemail: "<<students.email<<endl;

}

int main(){

    int size=5;

    Student \*students=new Student[size];

    for (int i=0; i<size; i++){

        cout<<"Student "<<i+1<<" :\n";

        input(students[i]);

    }

    cout<<"\nDisplaying all Details:\n";

    for (int i=0; i<size; i++){

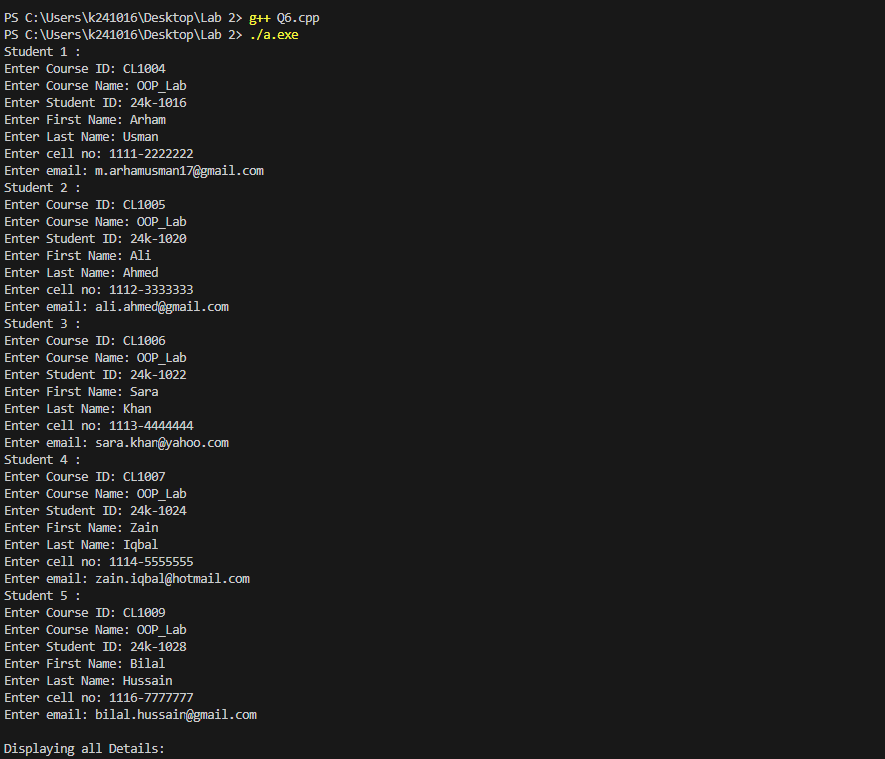
        cout<<"Student "<<i+1<<" :\n";

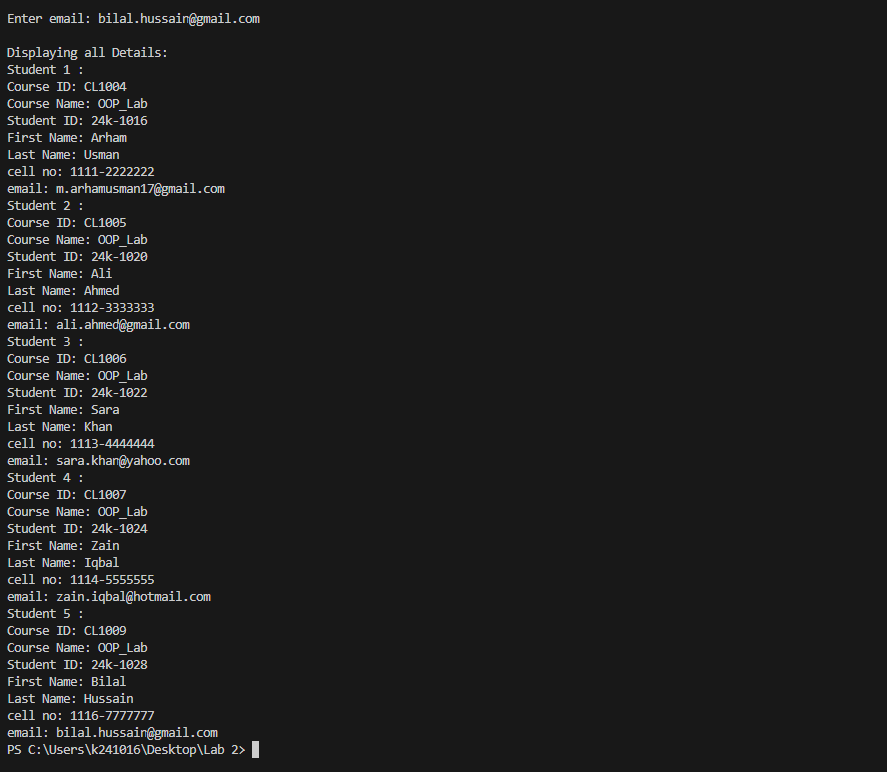
        display(students[i]);

    }

}

# OUTPUT:





QUESTION 7

# SOURCE CODE:

/\*

You are tasked with building a simple product management system for an online store.

1. Create a function that allows the addition of a new product to the system. The function should

take parameters such as product name, price, quantity, and any other relevant details.

2. Implement a function that takes a product ID as input and displays detailed information about the

product, including its name, price, quantity in stock, and any other relevant details.

3. Design a function that enables the update of product information. It should take a product ID as

well as the new details (e.g., updated price, quantity, etc.) and modify the existing product&#39;s

information accordingly.

4. Create a function that removes a product from the system based on its product ID. Ensure that

the inventory is updated after the removal.

\*/

#include <iostream>

using namespace std;

typedef struct{

    int id;

    string name;

    float price;

    int quantity;

}product;

void increase\_size(product \*&arr, int &size){

    product \*updated= new product[++size];

    for (int i=0; i<size-1; i++){

        updated[i]=arr[i];

    }

    delete[] arr;

    arr=updated;

}

void add\_product(product \*&arr, int id, string name, float price, int quantity){

    arr[id].id=id;

    arr[id].name=name;

    arr[id].price=price;

    arr[id].quantity=quantity;

}

void display(product \*&arr, int id){

    cout << "\n-----PRODUCT ID "<<arr[id].id<<"-----\n";

    cout<<"Name: "<<arr[id].name<<endl;

    cout<<"Price: "<<arr[id].price<<endl;

    cout<<"Quantity: "<<arr[id].quantity<<endl;

    cout<<endl;

}

void update(product \*&arr, int id, string name, float price, int quantity){

    arr[id].name=name;

    arr[id].price=price;

    arr[id].quantity=quantity;

}

void remove(product \*&arr, int id, int &size){

    size--;

    for (int i=id; i<size; i++){

        arr[i].name=arr[i+1].name;

        arr[i].price=arr[i+1].price;

        arr[i].quantity=arr[i+1].quantity;

    }

    product \*arr2=new product[size];

    for (int i=0; i<size; i++){

        arr2[i]=arr[i];

    }

    delete[] arr;

    arr=arr2;

}

void display\_all(product \*&arr, int size){

    for (int i=0; i<size; i++){

        if (arr[i].id!=-1){

            display(arr, i);

        }

    }

}

void menu(product \*&arr, int &num){

    int choice;

    int id;

    string name;

    float price;

    int quantity;

    while (true){

        cout<<"\nMenu:\nPress 1 to add a product\nPress 2 to view a product\n";

        cout<<"Press 3 to update a product\nPress 4 to remove a product\n";

        cout<<"Press 5 to view all inventory\nPress any other key to exit";

        cout<<"\nEnter your choice: ";

        fflush(stdout);

        cin>>choice;

        switch (choice){

            case 1:

                increase\_size(arr, num);

                cout<<"Product ID: "<<num-1<<endl;

                cout<<"Enter Name: ";

                fflush(stdout);

                cin>>name;

                cout<<"Enter price: ";

                fflush(stdout);

                cin>>price;

                cout<<"Enter quantity: ";

                fflush(stdout);

                cin>>quantity;

                add\_product(arr, num-1, name, price, quantity);

                break;

            case 2:

                cout<<"Enter Product ID: ";

                fflush(stdout);

                cin>>id;

                if (id>=num||id<0){

                    cout<<"Invalid ID!\n";

                }

                else{

                    display(arr, id);

                }

                break;

            case 3:

                cout<<"Enter Product ID: ";

                fflush(stdout);

                cin>>id;

                if (id>=num||id<0){

                    cout<<"Invalid ID!\n";

                }

                else{

                    cout<<"Enter Name: ";

                    fflush(stdout);

                    cin>>name;

                    cout<<"Enter price: ";

                    fflush(stdout);

                    cin>>price;

                    cout<<"Enter quantity: ";

                    fflush(stdout);

                    cin>>quantity;

                    update(arr, id, name, price, quantity);

                }

                break;

            case 4:

                cout<<"Enter Product ID: ";

                fflush(stdout);

                cin>>id;

                if (id>=num||id<0){

                    cout<<"Invalid ID!\n";

                }

                else{

                    remove(arr, id, num);

                }

                break;

            case 5:

                display\_all(arr, num);

                break;

            default:

                return;

        }

    }

}

int main(){

    product \*online\_store = new product[1];

    int number\_of\_products=0;

    cout<<"\n===PRODUCT MANAGEMENT SYSTEM===\n";

    menu(online\_store, number\_of\_products);

    delete[] online\_store;

}

# OUTPUT:

