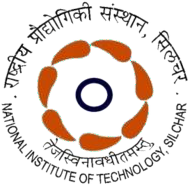
ASSIGNMENT I

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT NATIONAL INSTITUTE OF TECHNOLOGY SILCHAR ASSAM-788010, INDIA



Object Oriented Programming Lab

### 

### SUBMITTED BY:

**NAME:**TANVEER AHMED

**SCHOLAR ID**: 1815010

**SEC:**A

**Question 1**: Write a C++ program to input time in (MM:SS) format and add their values. Output should be in (HH:MM:SS) format.

**Instruction**: Add 3 values. Use a constructor to input the first two values and the third one should be a user input. Define a destructor to delete dynamic allocated memory.

**Input**: (MM:SS) should be in string format. Thereafter, use the MM and SS value to carry out the addition task.

**Output**: (HH:MM:SS) should be in string format.

**Code**:

#include <bits/stdc++.h>

using namespace std;

class Time

{

private:

int SS;

int MM;

int HH;

public:

Time()

{

SS = 0;

MM = 0;

HH = 0;

}

Time(int ch)

{

if (ch == 1)

{

SS = 20;

MM = 34;

HH = 0;

}

else

{

SS = 25;

MM = 46;

HH = 0;

}

}

Time(int mm, int ss)

{

SS = ss;

MM = mm;

HH = 0;

}

int returnTimeMM()

{

return MM;

}

int returnTimeSS()

{

return SS;

}

void displayTime()

{

cout << to\_string(HH) + ":" + to\_string(MM) + ":" + to\_string(SS);

}

void addTime(Time T)

{

MM += T.returnTimeMM();

SS += T.returnTimeSS();

if (SS >= 60)

{

MM++;

SS = SS%60;

}

if (MM >= 60)

{

HH++;

MM = MM%60;

}

}

~Time()

{

cout << "\nDDD\n";

}

} T1(1), T2(2);

int main()

{

string tm3, tm4;

//Displaying Time1, Time2

cout << "\nTIME\_1:- "; T1.displayTime();

cout << "\nTIME\_2:- "; T2.displayTime();

cout << "\nEnter TIME\_3: ";

cin >> tm3;

//Parsing to get Time3 and Displaying

vector <string> time;

stringstream check1(tm3);

string intermediate;

while(getline(check1, intermediate, ':'))

time.push\_back(intermediate);

Time T3(stoi(time[0]), stoi(time[1]));

cout << "\nTIME\_3:- "; T3.displayTime();

//Addition

Time T4;

T4.addTime(T1);

T4.addTime(T2);

T4.addTime(T3);

//Displaying Result

cout << "\nREQUIRED ADDED TIME:- "; T4.displayTime();

return 0;

}

Outputs

Test Case 1:

TIME\_1:- 0:44:50

TIME\_2:- 0:36:15

Enter TIME\_3: 21:12

TIME\_3:- 0:21:12

DDD

DDD

DDD

REQUIRED ADDED TIME: - 1:42:17

DDD

DDD  
DDD

---------------x------------------x--------------------x-------------------x-------------------x-------------

**Question 2**: (Operator Overloading) Write a C++ program to add two distances using binary plus (+) operator overloading.

**Input**: (FF-IN) should be in string format. Thereafter, use the mm and dd value to carry out the addition task.

**Output**: (FF-IN) should be in string format.

**Code**:

#include <bits/stdc++.h>

using namespace std;

class Distance

{

private:

int FF;

int IN;

public:

Distance()

{

FF = 0;

IN = 0;

}

Distance(int ff, int in)

{

FF = ff;

IN = in;

}

int returnFF()

{

return FF;

}

int returnIN()

{

return IN;

}

void displayDistance()

{

cout << to\_string(FF) + "-" + to\_string(IN);

}

Distance operator + (Distance const &obj)

{

Distance D;

D.IN = IN + obj.IN;

D.FF = FF + obj.FF;

if (D.IN >= 12)

{

D.FF++;

D.IN = D.IN%12;

}

return D;

}

};

int main()

{

string dist1, dist2, distance;

//Input Distance1, Distance2

cout << "\nEnter Following Distances in (Feet-Inches) format: ";

cout << "\n#1\n\tEnter Distance: ";

cin >> dist1;

cout << "\n#2\n\tEnter Distance: ";

cin >> dist2;

//Parsing to get Distance1

vector <string> distA;

stringstream check1(dist1);

string intermediate1;

while(getline(check1, intermediate1, '-'))

distA.push\_back(intermediate1);

Distance D1(stoi(distA[0]), stoi(distA[1]));

cout << "\nDistance\_1: "; D1.displayDistance();

//Parsing to get Distance2

vector <string> distB;

stringstream check2(dist2);

string intermediate2;

while(getline(check2, intermediate2, '-'))

distB.push\_back(intermediate2);

Distance D2(stoi(distB[0]), stoi(distB[1]));

cout << "\nDistance\_2: "; D2.displayDistance();

//Addition

Distance D = D1 + D2;

//Displaying Result

cout << "\n\nREQUIRED FINAL DISTANCE: "; D.displayDistance();

cout << "\n\n";

return 0;

}

OUTPUT

Enter Following Distances in (Feet-Inches) format:

#1

Enter Distance: 5-3

#2

Enter Distance: 5-10

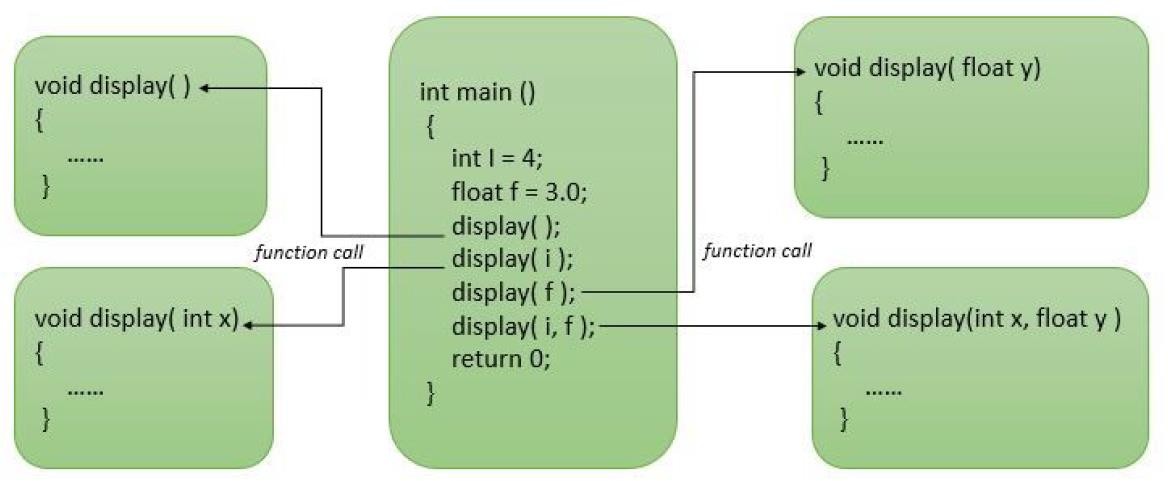
Distance\_1: 5-3

Distance\_2: 5-10

REQUIRED FINAL DISTANCE: 11-1

---------------x------------------x--------------------x-------------------x-------------------x-------------

**Question 3**: (Function Overloading) Write a C++ program to implement the following:



**Code:**

#include <bits/stdc++.h>

using namespace std;

void display()

{

cout << "\nNOTHING";

}

void display(int i)

{

cout << "\ni = " << i;

}

void display(float f)

{

cout << "\nf = " << f;

}

void display(int i, float f)

{

cout << "\ni = " << i << ", f = " << f << "\n\n";

}

int main()

{

int i = 4;

float f = 3.0;

display();

display(i);

display(f);

display(i, f);

return 0;

}

**OUTPUT**

TESTCASE 1:

NOTHING

i = 5

f = 4

i = 5, f = 4

---------------x------------------x--------------------x-------------------x-------------------x-------------

**Question 4**: (File Handling) Write a C++ program to write and read objects in the file using read and write functions.

#include <bits/stdc++.h>

using namespace std;

ofstream SaveFiles;

ifstream ReadSaveFiles;

class BioData

{

private:

int slno;

int id;

int semester;

public:

BioData()

{

slno = 0;

id = 69;

semester = 4;

}

BioData(int sl, int ID, int sem)

{

slno = sl;

id = ID;

semester = sem;

}

int AccessSlno(int password)

{

if (password == 132)

return slno;

}

void DisplayBioData(int password)

{

if (password == 132)

{

cout << "\n\nSTUDENT #" << slno << ": ";

cout << "\n\tSCHOLAR ID: " << id;

cout << "\n\tSEMESTER: " << semester;

cout << "\n";

}

}

} ModelStudent;

void READ()

{

int ctr = 0;

ReadSaveFiles.open("SAVDAT", ios::in | ios::binary);

ReadSaveFiles.seekg(0, ios::beg);

while(!ReadSaveFiles.eof())

{

ReadSaveFiles.read((char \*)&ModelStudent, sizeof(ModelStudent));

ctr++;

}

ctr--;

ReadSaveFiles.close();

ReadSaveFiles.open("SAVDAT", ios::in | ios::binary);

ReadSaveFiles.seekg(0, ios::beg);

while(ctr--)

{

ReadSaveFiles.read((char \*)&ModelStudent, sizeof(ModelStudent));

ModelStudent.DisplayBioData(132);

}

ReadSaveFiles.close();

}

void WRITE()

{

//Accessing File and Calculating Silo No. for Writing

SaveFiles.open("SAVDAT", ios::app | ios::binary);

ReadSaveFiles.open("SAVDAT", ios::in | ios::binary);

SaveFiles.seekp(0, ios::end);

int loc = SaveFiles.tellp();

//cout<<" TELLP()=" << loc;

int sl = (loc/sizeof(ModelStudent)) + 1;

//Collecting Data for Writing into SAVDAT

int ID, sem;

cout << "\nEnter Scholar ID: "; cin >> ID;

cout << "Enter Semester: "; cin >> sem;

//Creating Required Object for Data Entry

BioData Student(sl, ID, sem);

SaveFiles.write((char \*)&Student, sizeof(Student));

ReadSaveFiles.seekg(loc);

ReadSaveFiles.read((char \*)&Student, sizeof(Student));

cout<<"\nFile no. #"<<Student.AccessSlno(132)<<" Created! ";

Student.DisplayBioData(132);

//test

//cout << "\n\n\t\tSIZE\_OF\_STUDENT: " << sizeof(Student);

//cout << "\n\n\t\tSIZE\_OF\_MODELSTUDENT: " << sizeof(ModelStudent);

//cout << "\n\n\t\tSIZE\_OF\_STRING: " << sizeof(string);

//cout << "\n\n\t\tSIZE\_OF\_INT: " << sizeof(int);

//cout << "\n\n";

//Closing Opened Streams

ReadSaveFiles.close();

SaveFiles.close();

}

int main()

{

int choice = 0;

BEG:

cout << "\n\n1.READ\n2.WRITE\n3.EXIT\nCHOOSE: "; cin >> choice;

if (choice == 1)

READ();

else if (choice == 2)

WRITE();

else

goto END;

goto BEG;

END:

return 0;

}

**OUTPUT**

1.READ

2.WRITE

3.EXIT

CHOOSE: 2

Enter Scholar ID: 1815010

Enter Semester: 4

File no. #1 Created!

STUDENT #1:

SCHOLAR ID: 1815010

SEMESTER: 4

1.READ

2.WRITE

3.EXIT

CHOOSE: 2

Enter Scholar ID: 1715010

Enter Semester: 6

File no. #2 Created!

STUDENT #2:

SCHOLAR ID: 1715010

SEMESTER: 6

1.READ

2.WRITE

3.EXIT

CHOOSE: 1

STUDENT #1:

SCHOLAR ID: 1815010

SEMESTER: 4

STUDENT #2:

SCHOLAR ID: 1715010

SEMESTER: 6

---------------x------------------x--------------------x-------------------x-------------------x-------------

**Question 5**:

(File Handling) Write a C++ program to read a file to write the following output in a new file:

a.Number of characters present in the input file.

b.Number of unique present in the input file.

c.Occurrence of each character in the input file. Use separate functions to execute the three tasks.

**Code:**

#include <bits/stdc++.h>

using namespace std;

ifstream inFile;

ofstream outFile;

int CountUniqueCharacters(char\* str, int n)

{

int count = 0;

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

bool appears = false;

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

if (str[j] == str[i]){

appears = true;

break;

}

}

if (!appears){

count++;

}

}

return count;

}

void printCharWithFreq(char\* str, int n)

{

int freq[1000];

memset(freq, 0, sizeof(freq));

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

freq[(int)str[i]]++;

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

{

if (freq[(int)str[i]] != 0)

{

outFile << str[i] << ": " << freq[(int)str[i]] << "\n";

freq[(int)str[i]] = 0;

}

}

}

int main()

{

char text[1000];

int i = 0;

inFile.open("Characters.txt", ios::in);

outFile.open("Characters\_Analysis.txt", ios::out);

while (!inFile.eof())

inFile >> text[i++];

i--;

outFile << "Characters: " << i << "\n";

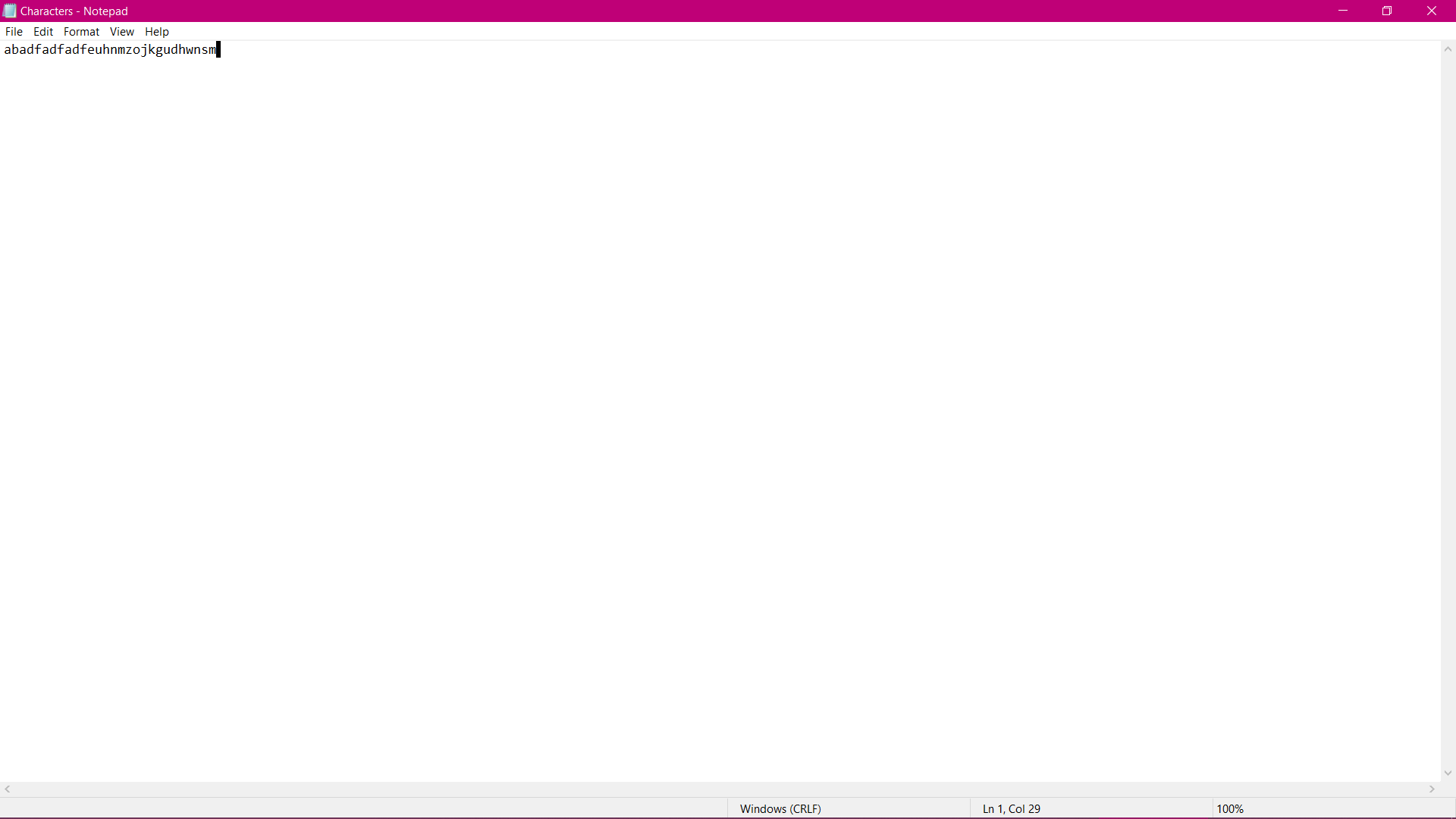
outFile << "Uniques: " << CountUniqueCharacters(text, i) << "\n";

printCharWithFreq(text, i);

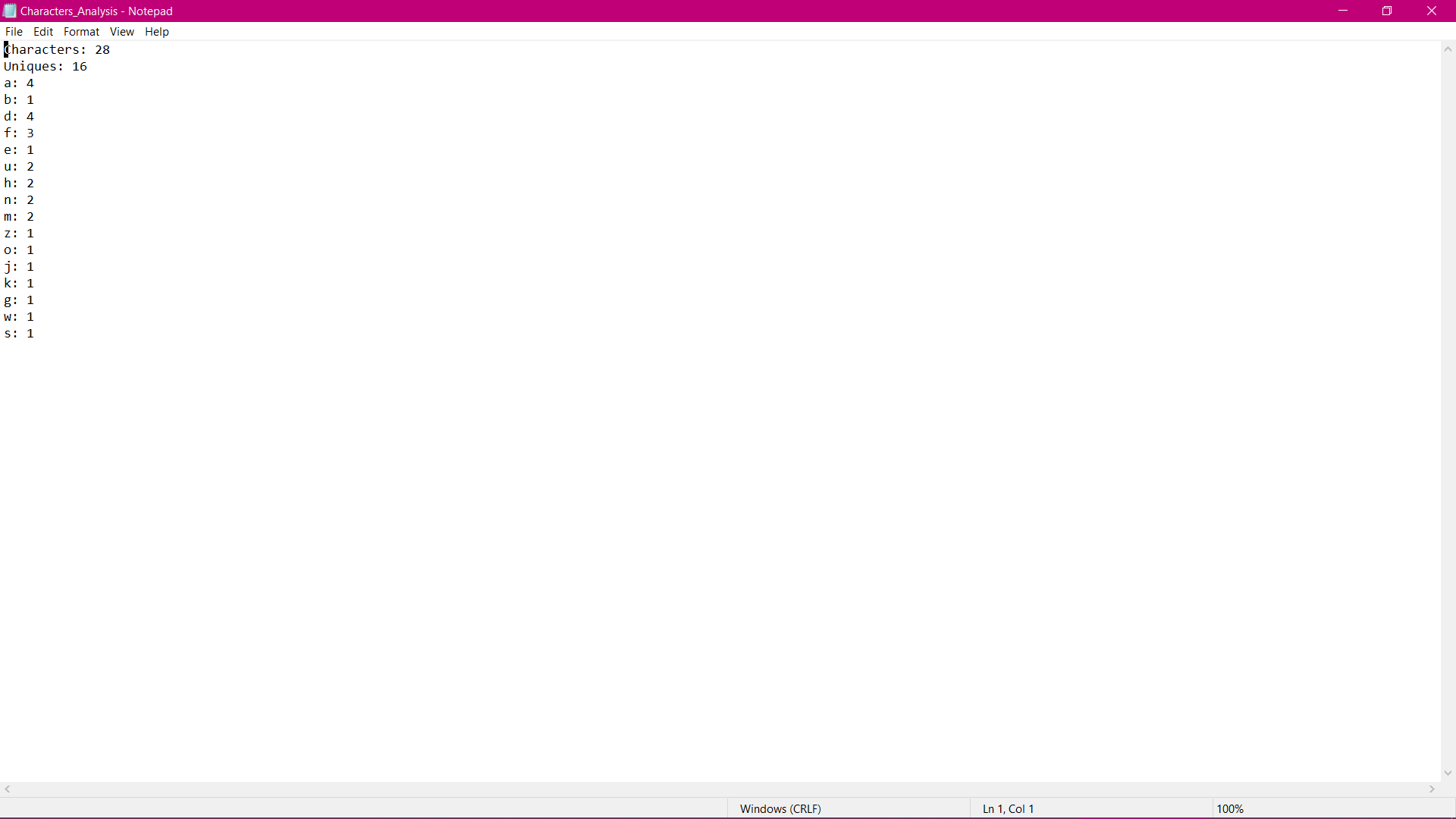
return 0;

}

INPUT



OUTPUT



---------------x------------------x--------------------x-------------------x-------------------x-------------

**Question 6**: (Template) Write a C++ program to define a complex number using a template. The real and imaginary value are always of different data types. Use constructors to assign value to real and imaginary parts of complex numbers. Extend the program to accomplish the operations:

a.Addition of two complex number

b.Subtract of two complex number

c.A display function to show the complex number

Code:

#include <bits/stdc++.h>

using namespace std;

template <class T, class V>

class ComplexNumber

{

private:

T real;

V imaginary;

public:

ComplexNumber()

{

real = 0.0;

imaginary = 0.0;

}

ComplexNumber(T rl, V im)

{

real = rl;

imaginary = im;

}

void Display()

{

cout << ": " << real << " + i" << imaginary << "\n";

}

ComplexNumber<T&, V&> operator = (const ComplexNumber&);

ComplexNumber<T, V> operator + (const ComplexNumber&)const;

ComplexNumber<T, V> operator - (const ComplexNumber&)const;

};

//Complex Number Copy Constructor

template <class T, class V>

ComplexNumber<T&, V&> ComplexNumber<T, V>::operator = (const ComplexNumber& other)

{

if (this != &other)

{

real = other.real;

imaginary = other.imaginary;

}

return \*this;

}

//Overloaded Addition Operator

template<class T, class V>

ComplexNumber<T, V> ComplexNumber<T, V>::operator +(const ComplexNumber<T, V>& other) const

{

ComplexNumber<T, V> temp;

temp.real = real + other.real;

temp.imaginary = imaginary + other.imaginary;

return temp;

}

//Overloaded Subtraction Operator

template<class T, class V>

ComplexNumber<T, V> ComplexNumber<T, V>::operator - (const ComplexNumber<T, V>& other) const

{

ComplexNumber<T, V> temp;

temp.real = real - other.real;

temp.imaginary = imaginary - other.imaginary;

return temp;

}

int main()

{

int choice = 0;

float R1, R2;

long double I1, I2;

//Initializing Complex Numbers

cout << "\nComplexNumber #1: ";

cout << "\n\tREAL: "; cin >> R1;

cout << "\tIMAGINARY: "; cin >> I1;

cout << "\nComplexNumber #2: ";

cout << "\n\tREAL: "; cin >> R2;

cout << "\tIMAGINARY: "; cin >> I2;

ComplexNumber<float, long double> C1(R1, I1), C2(R2, I2);

//Display

//cout << "\nThe two Imaginary Numbers are: \n";

//C1.Display();

//C2.Display();

MENU:

cout << "\n1. ADD";

cout << "\n2. SUBTRACT";

cout << "\n3. DISPLAY";

cout << "\nELSE. EXIT";

cout << "\nCHOOSE: "; cin >> choice;

if (choice == 1) {

ComplexNumber<float, long double> C = C1 + C2;

cout << "\n";

C.Display();

goto MENU;

}

if (choice == 2) {

ComplexNumber<float, long double> c = C2 - C1;

cout << "\n";

c.Display();

goto MENU;

}

if (choice == 3) {

cout << "\nThe two Imaginary Numbers are: \n";

C1.Display();

C2.Display();

goto MENU;

}

else

goto END;

END:

return 0;

}

**OUTPUT**

ComplexNumber #1:

REAL: 2

IMAGINARY: -3

ComplexNumber #2:

REAL: 3

IMAGINARY: 4

1. ADD

2. SUBTRACT

3. DISPLAY

ELSE. EXIT

CHOOSE: 1

: 5 + i1

1. ADD

2. SUBTRACT

3. DISPLAY

ELSE. EXIT

CHOOSE: 2

: 1 + i7

1. ADD

2. SUBTRACT

3. DISPLAY

ELSE. EXIT

CHOOSE: 3

The two Imaginary Numbers are:

: 2 + i-3

: 3 + i4

1. ADD

2. SUBTRACT

3. DISPLAY

ELSE. EXIT

CHOOSE: 6

---------------x------------------x--------------------x-------------------x-------------------x-------------

**Question 7**: (Exception Handling) WAP in C++ to achieve following

a.A function to read two float type numbers from keyboard

b.A function to calculate the division of these two numbers

c.A try block to throw an exception when a wrong type of data is keyed in

d.A try block to detect and throw an exception if the condition “divide-by-zero” occurs

e.Appropriate catch block to handle the exceptions thrown

**Code:**

#include <bits/stdc++.h>

#include <conio.h>

using namespace std;

#define NO 2

char kp;

int index = 0;

string flotus[100];

string convertToString(char\* a, int size)

{

int i;

string s = "";

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

s = s + a[i];

}

return s;

}

int isFloat(string str)

{

int i = 0, j = str.length() - 1;

// Handling whitespaces

while (i < str.length() && str[i] == ' ')

i++;

while (j >= 0 && str[j] == ' ')

j--;

if (i > j)

return 0;

// if string is of length 1 and the only

// character is not a digit

if (i == j && !(str[i] >= '0' && str[i] <= '9'))

return 0;

// If the 1st char is not '+', '-', '.' or digit

if (str[i] != '.' && str[i] != '+'

&& str[i] != '-' && !(str[i] >= '0' && str[i] <= '9'))

return 0;

// To check if a '.' or 'e' is found in given

// string. We use this flag to make sure that

// either of them appear only once.

bool flagDotOrE = false;

for (i; i <= j; i++) {

// If any of the char does not belong to

// {digit, +, -, ., e}

if (str[i] != 'e' && str[i] != '.'

&& str[i] != '+' && str[i] != '-'

&& !(str[i] >= '0' && str[i] <= '9'))

return 0;

if (str[i] == '.') {

// checks if the char 'e' has already

// occurred before '.' If yes, return 0.

if (flagDotOrE == true)

return 0;

// If '.' is the last character.

if (i + 1 > str.length())

return 0;

// if '.' is not followed by a digit.

if (!(str[i + 1] >= '0' && str[i + 1] <= '9'))

return 0;

}

else if (str[i] == 'e') {

// set flagDotOrE = 1 when e is encountered.

flagDotOrE = true;

// if there is no digit before 'e'.

if (!(str[i - 1] >= '0' && str[i - 1] <= '9'))

return 0;

// If 'e' is the last Character

if (i + 1 > str.length())

return 0;

// if e is not followed either by

// '+', '-' or a digit

if (str[i + 1] != '+' && str[i + 1] != '-'

&& (str[i + 1] >= '0' && str[i] <= '9'))

return 0;

}

}

/\* If the string skips all above cases, then

it is numeric\*/

return 1;

}

void ReadFloat()

{

int ctr = 0, i = 0, j = 0;

char A[100];

//UI

cout << "\nEnter Two Floating point Numbers: \n";

//KeyPress Detection

while (ctr < NO)

{

kp = getch();

cout << kp;

A[i++] = kp;

int k = (int)kp;

if (k == 13) {

flotus[index++] = convertToString(A, i-1);

ctr++;

i = 0;

cout << "\n";

}

}

}

float divideFloat(float f1, float f2)

{

if (f2 == 0) {

throw runtime\_error("Math error: Attempted to divide by Zero\n");

}

return f1 / f2;

}

int main()

{

bool bf1, bf2;

float divFloat;

//DATA ENTRY

ReadFloat();

//cout << "\n" << flotus[0];

//cout << "\n" << flotus[1];

//Data Type Analysis - Exception Handling

try

{

//cout << "\nflt " << isFloat(flotus[0]);

if (!isFloat(flotus[0]))

throw flotus[0];

if (!isFloat(flotus[1]))

throw flotus[1];

}

catch(string F)

{

cout << "\nException occurred";

cout << "\nWRONG DATA TYPE.....EXITING\n";

exit(0);

}

//DIVISION

try {

divFloat = divideFloat (stof(flotus[0]), stof(flotus[1]));

}

//Divide by Zero - Exception Handling

catch (runtime\_error& e) {

cout << "Exception occurred" << "\n" << e.what();

exit(0);

}

//Result

cout << "\nRequired Result after Division of Floating points: " << divFloat << "\n\n";

return 0;

}

**OUTPUT**

TESTCASE 1:

Enter Two Floating point Numbers:

21.1

22.3

Required Result after Division of Floating points: 0.946188

TESTCASE 2:

Enter Two Floating point Numbers:

2f

6.9

Exception occurred

WRONG DATA TYPE.....EXITING

---------------x------------------x--------------------x-------------------x-------------------x-------------

**Question 8:**

(Standard Template Library) WAP in C++ that accepts a shopping list of five items from the keyboard and stores them in a vector.

Extend the program to accomplish the following:

a.Use find() algorithm to locate the position of a specified value

b.To add an item at a specified location

c.To add an item at the end

d.To print the contents of the vectors

e.Use count() algorithm to count the number of elements

f.To delete a specified item in the list

**Code:**

#include <bits/stdc++.h>

using namespace std;

int menu()

{

int ch = 0;

cout << "\n\nINSTRUCTION MANUAL: ";

cout << "\n1. Find item in the Shopping List.";

cout << "\n2. Add item at specified position in the Shopping List.";

cout << "\n3. Add an item at the end of the Shopping List.";

cout << "\n4. Print the Shopping List.";

cout << "\n5. Count no. of items in the Shopping List.";

cout << "\n6. Delete an item from the Shopping List.";

cout << "\nELSE. EXIT.";

cout << "\nCHOOSE: "; cin >> ch;

return ch;

}

int main()

{

int ch = 0, ps;

string item, key;

vector<string> shopping\_list;

vector<string>::iterator pos;

//Initial Cart Entry

cout << "\nEnter 5 items in the Shopping List: \n";

for (int i = 0; i < 5; i++)

{

cout << "\tITEM #" << i+1 << ": ";

cin >> item;

shopping\_list.push\_back(item);

}

MENU:

ch = menu();

CHOICES:

if (ch == 1)

{

cout << "\nEnter item name to find: ";

cin >> key;

pos = find(shopping\_list.begin(), shopping\_list.end(), key);

if (pos != shopping\_list.end())

{

cout << "item " << key <<" found at position : " ;

cout << pos - shopping\_list.begin() + 1 << " in the Shopping List\n\n";

}

else

cout << "\nitem not found.\n\n";

goto MENU;

}

if (ch == 2)

{

cout << "\nEnter item to add to the Shopping List: ";

cin >> key;

cout << "Enter position of entry in the Shopping List: ";

cin >> ps;

shopping\_list.insert(shopping\_list.begin() + ps - 1, key);

goto MENU;

}

if (ch == 3)

{

cout << "\nEnter item to add to the Shopping List: ";

cin >> key;

shopping\_list.push\_back(key);

goto MENU;

}

if (ch == 4)

{

cout << "\nFINAL SHOPPING LIST: ";

for (int j = 0; j < shopping\_list.size(); j++)

cout << "\n" << j+1 << ". " << shopping\_list.at(j);

goto MENU;

}

if (ch == 5)

{

int cnt = 0;

for (int a = 0; a < shopping\_list.size(); a++)

cnt += count(shopping\_list.begin(), shopping\_list.end(), shopping\_list.at(a));

cout << "\nNo. of items in this Shopping List: " << cnt << "\n";

goto MENU;

}

if (ch == 6)

{

cout << "Enter position of deletion in the Shopping List: ";

cin >> ps;

shopping\_list.erase(shopping\_list.begin() + ps - 1);

goto MENU;

}

else

{

exit(0);

}

END:

return 0;

}

OUTPUT

Enter 5 items in the Shopping List:

ITEM #1: chips

ITEM #2: mayo

ITEM #3: egg

ITEM #4: pencil

ITEM #5: laptop

INSTRUCTION MANUAL:

1. Find item in the Shopping List.

2. Add item at specified position in the Shopping List.

3. Add an item at the end of the Shopping List.

4. Print the Shopping List.

5. Count no. of items in the Shopping List.

6. Delete an item from the Shopping List.

ELSE. EXIT.

CHOOSE: 1

Enter item name to find: laptop

item laptop found at position : 5 in the Shopping List

INSTRUCTION MANUAL:

1. Find item in the Shopping List.

2. Add item at specified position in the Shopping List.

3. Add an item at the end of the Shopping List.

4. Print the Shopping List.

5. Count no. of items in the Shopping List.

6. Delete an item from the Shopping List.

ELSE. EXIT.

CHOOSE: 2

Enter item to add to the Shopping List: hairdryer

Enter position of entry in the Shopping List: 2

INSTRUCTION MANUAL:

1. Find item in the Shopping List.

2. Add item at specified position in the Shopping List.

3. Add an item at the end of the Shopping List.

4. Print the Shopping List.

5. Count no. of items in the Shopping List.

6. Delete an item from the Shopping List.

ELSE. EXIT.

CHOOSE: 4

FINAL SHOPPING LIST:

1. chips

2. hairdryer

3. mayo

4. egg

5. pencil

6. laptop

INSTRUCTION MANUAL:

1. Find item in the Shopping List.

2. Add item at specified position in the Shopping List.

3. Add an item at the end of the Shopping List.

4. Print the Shopping List.

5. Count no. of items in the Shopping List.

6. Delete an item from the Shopping List.

ELSE. EXIT.

CHOOSE: 6

Enter position of deletion in the Shopping List: 3

INSTRUCTION MANUAL:

1. Find item in the Shopping List.

2. Add item at specified position in the Shopping List.

3. Add an item at the end of the Shopping List.

4. Print the Shopping List.

5. Count no. of items in the Shopping List.

6. Delete an item from the Shopping List.

ELSE. EXIT.

CHOOSE: 4

FINAL SHOPPING LIST:

1. chips

2. hairdryer

3. egg

4. pencil

5. laptop

INSTRUCTION MANUAL:

1. Find item in the Shopping List.

2. Add item at specified position in the Shopping List.

3. Add an item at the end of the Shopping List.

4. Print the Shopping List.

5. Count no. of items in the Shopping List.

6. Delete an item from the Shopping List.

ELSE. EXIT.

CHOOSE: 5

No. of items in this Shopping List: 5

INSTRUCTION MANUAL:

1. Find item in the Shopping List.

2. Add item at specified position in the Shopping List.

3. Add an item at the end of the Shopping List.

4. Print the Shopping List.

5. Count no. of items in the Shopping List.

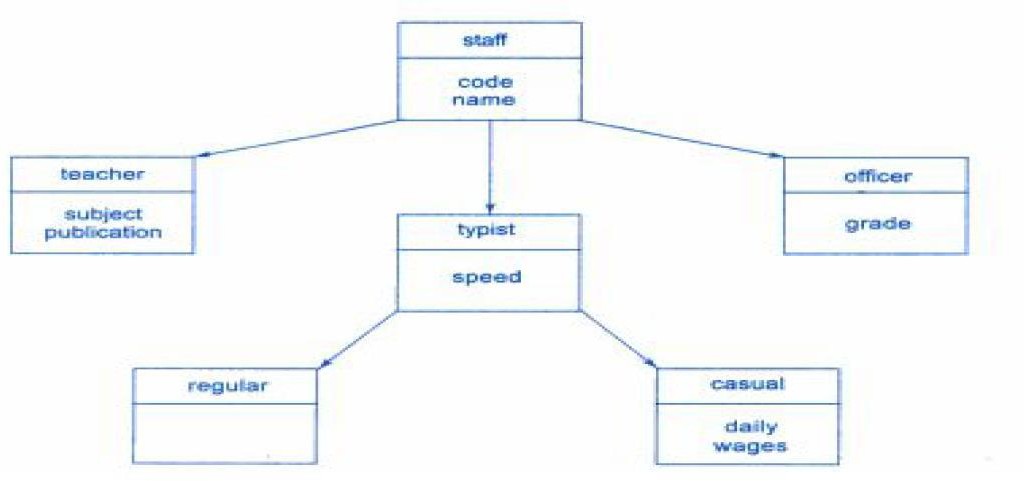
6. Delete an item from the Shopping List.

ELSE. EXIT.

CHOOSE: 7

---------------x------------------x--------------------x-------------------x-------------------x-------------

**Question 9:** (Inheritance) An educational institution wishes to maintain a database of its employees. The database is divided into a number of classes whose hierarchical relationships are shown in the following figure. The figure also shows the minimum information required for each class. Specify all classes and define functions to create the database and retrieve individual information as and when required.



**Code:**

#include <bits/stdc++.h>

using namespace std;

int t = 0, o = 0, c = 0, r = 0;

class staff

{

public:

int code;

string name;

};

class teacher : public staff

{

public:

string subject;

string publication;

void TDisplay()

{

cout << "\nCODE: " << code;

cout << "\nNAME: " << name;

cout << "\nSUBJECT: " << subject;

cout << "\nPUBLICATION: " << publication;

}

} T[100];

class typist : public staff

{

public:

int speed;

};

class officer : public staff

{

public:

int grade;

void ODisplay()

{

cout << "\nCODE: " << code;

cout << "\nNAME: " << name;

cout << "\nGRADE: " << grade;

}

} O[100];

class regular : public typist

{

public:

void RDisplay()

{

cout << "\nCODE: " << code;

cout << "\nNAME: " << name;

cout << "\nSPEED: " << speed;

}

} R[100];

class casual : public typist

{

public:

int daily\_wages;

void CDisplay()

{

cout << "\nCODE: " << code;

cout << "\nNAME: " << name;

cout << "\nSPEED: " << speed;

cout << "\nDAILY WAGES: " << daily\_wages;

}

} C[100];

int main()

{

int ch = 0, choice = 0;

INIT:

cout << "\n\nINSTRUCTION: ";

cout << "\n1. Add Entry";

cout << "\n2. Find Entry";

cout << "\nELSE. EXIT";

cout << "\nCHOOSE: "; cin >> ch;

if (ch == 1)

{

cout << "\nWHAT ARE YOU?: ";

cout << "\n1. Teacher";

cout << "\n2. Officer";

cout << "\n3. Regular Typist";

cout << "\n4. Casual Typist";

cout << "\nCHOOSE: "; cin >> choice;

if (choice == 1)

{

cout << "\nCODE: "; cin >> T[t].code;

cout << "NAME: "; cin >> T[t].name;

cout << "SUBJECT: "; cin >> T[t].subject;

cout << "PUBLICATION: "; cin >> T[t++].publication;

}

else if (choice == 2)

{

cout << "\nCODE: "; cin >> O[o].code;

cout << "NAME: "; cin >> O[o].name;

cout << "GRADE: "; cin >> O[o++].grade;

}

else if (choice == 3)

{

cout << "\nCODE: "; cin >> R[r].code;

cout << "NAME: "; cin >> R[r].name;

cout << "SPEED: "; cin >> R[r++].speed;

}

else if (choice == 4)

{

cout << "\nCODE: "; cin >> C[c].code;

cout << "NAME: "; cin >> C[c].name;

cout << "SPEED: "; cin >> C[c].speed;

cout << "DAILY WAGES: "; cin >> C[c++].daily\_wages;

}

goto INIT;

}

if (ch == 2)

{

int CODE, ctr = 0, ctr\_type = 0, a = 0, b = 0, e = 0, d = 0;

cout << "\nEnter CODE of Staff you need to find: ";

cin >> CODE;

for (a = 0; a < t; a++)

{

if (T[a].code == CODE) {

ctr = 1;

ctr\_type = 1;

goto END\_LOOP;

}

}

for (b = 0; b < o; b++)

{

if (O[b].code == CODE) {

ctr = 1;

ctr\_type = 2;

goto END\_LOOP;

}

}

for (e = 0; e < r; e++)

{

if (R[e].code == CODE) {

ctr = 1;

ctr\_type = 3;

goto END\_LOOP;

}

}

for (d = 0; d < c; d++)

{

if (C[d].code == CODE) {

ctr = 1;

ctr\_type = 4;

goto END\_LOOP;

}

}

END\_LOOP:

if (ctr == 0) {

cout << "\nNO SUCH STAFF MEMBER FOUND.....\n";

goto INIT;

}

else

{

if (ctr\_type == 1)

{

cout << "\nTEACHER: ";

T[a].TDisplay();

cout << "\n\n";

goto INIT;

}

if (ctr\_type == 2)

{

cout << "\nOFFICER: ";

O[b].ODisplay();

cout << "\n\n";

goto INIT;

}

if (ctr\_type == 3)

{

cout << "\nREGULAR TYPIST: ";

R[e].RDisplay();

cout << "\n\n";

goto INIT;

}

if (ctr\_type == 4)

{

cout << "\nCASUAL TYPIST: ";

C[d].CDisplay();

cout << "\n\n";

goto INIT;

}

}

}

else

goto END;

END:

return 0;

}

**OUTPUT**

INSTRUCTION:

1. Add Entry

2. Find Entry

ELSE. EXIT

CHOOSE: 1

WHAT ARE YOU?:

1. Teacher

2. Officer

3. Regular Typist

4. Casual Typist

CHOOSE: 4

CODE: 10

NAME: tanveer

SPEED: 60

DAILY WAGES: 500

INSTRUCTION:

1. Add Entry

2. Find Entry

ELSE. EXIT

CHOOSE: 1

WHAT ARE YOU?:

1. Teacher

2. Officer

3. Regular Typist

4. Casual Typist

CHOOSE: 1

CODE: 1

NAME: ramu

SUBJECT: physics

PUBLICATION: gorgore

INSTRUCTION:

1. Add Entry

2. Find Entry

ELSE. EXIT

CHOOSE: 1

WHAT ARE YOU?:

1. Teacher

2. Officer

3. Regular Typist

4. Casual Typist

CHOOSE: 2

CODE: 12

NAME: rawat

GRADE: 2

INSTRUCTION:

1. Add Entry

2. Find Entry

ELSE. EXIT

CHOOSE: 2

Enter CODE of Staff you need to find: 12

OFFICER:

CODE: 12

NAME: rawat

GRADE: 2

INSTRUCTION:

1. Add Entry

2. Find Entry

ELSE. EXIT

CHOOSE: 2

Enter CODE of Staff you need to find: 10

CASUAL TYPIST:

CODE: 10

NAME: tanveer

SPEED: 60

DAILY WAGES: 500

INSTRUCTION:

1. Add Entry

2. Find Entry

ELSE. EXIT

CHOOSE: 8

---------------x------------------x--------------------x-------------------x-------------------x-------------

**Question 10:** (Pointers and Virtual Class)

Create a base class called shape. Use this class to store two double type values that could be used to compute the area of figures. Derive two specific classes called triangle and rectangle from the base shape. Add to the base class, a member function get\_data() to initialize base class data members and another member function display\_area() to compute and display the area of figures. Make display\_area() as a virtual function and redefine this function in the derived classes to suit their requirements. Using these three classes, design a program that will accept dimensions of a triangle or a rectangle interactively, and display the area. Remember the two values given as input will be treated as lengths of two sides in the case of rectangles and as base and height in the case of triangles. Use pointers to access the class.

**Code:**

#include <bits/stdc++.h>

using namespace std;

class Shape

{

protected:

double l,b;

public:

void getdata(double l1,double b1)

{

l=l1;

b=b1;

}

void virtual display\_area()=0;

};

class triangle:public Shape

{

public:

void display\_area()

{

cout<<"\nArea of Triangle:"<<(0.5\*l\*b);

}

};

class Rectangle:public Shape

{

public:

void display\_area()

{

cout<<"\nArea of Rectangle:"<<l\*b;

}

};

void main()

{

Shape \* s1;

triangle t1;

s1=&t1;

s1->getdata(20,10);

s1->display\_area();

Rectangle r1;

s1=&r1;

s1->getdata(5,4);

s1->display\_area();

}

OUTPUT

Area of Triangle:400

Area of Rectangle:24

---------------x------------------x--------------------x-------------------x-------------------x-------------

---------------x------------------x--------------------x-------------------x-------------------x-------------