**QUESTION 1**

**Write a C++ program taking employee name.Employee id and current salary.Define a function where program asks for increment or decrement .Based on this decide the status,If promoted or demoted.Cutoff for promotion salary is greater than or equal to 50000**

**Program**

**#include <iostream>**

**#include <string>**

**#include <iomanip>**

**using namespace std;**

**class employee {**

**private :**

**int code ;**

**string name ;**

**float original\_salary;**

**float incremented\_salary ;**

**public :**

**void input(){**

**cout<<"Enter the five digit employee code\n";**

**cin>>code ;**

**cin.ignore() ;**

**cout<<"Enter the name of the employee\n";**

**getline(cin,name) ;**

**cout <<"Enter the original salary of the employee\n" ;**

**cin>>original\_salary ;**

**return ;**

**}**

**int perks(){**

**int upd,n ;**

**int sal = original\_salary;**

**cout << "Enter number of perks \n ";**

**cin>>n ;**

**for (int i=1;i<=n;i++){**

**cout <<"Perk "<<i<<"\n Enter the salary hike/decrease given by the company in Rs:";**

**cin>>upd ;**

**sal +=upd ;**

**}incremented\_salary = sal ;**

**return 0 ;**

**}**

**int status(){**

**if(incremented\_salary>=50000){**

**return 1 ;**

**}**

**return 0 ;**

**}**

**void display(string state){**

**cout<<setw(6)<<code<<setw(25)<<name<<setw(17)<<original\_salary <<setw(17)<< incremented\_salary<<setw(10) <<state <<endl ;**

**}**

**};**

**int main (){**

**int n ;**

**string state[100] ;**

**employee s[100];**

**cout<<"Enter the number of employees\n";**

**cin>>n ;**

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

**s[i].input() ;**

**s[i].perks() ;**

**if(s[i].status()){**

**state[i] = "Promoted" ;**

**}else {**

**state[i] = "Demoted" ;**

**}**

**}**

**cout << "\nEmployee Code   Employee Name    Original Salary   Incremented Salary   Status"<<endl ;**

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

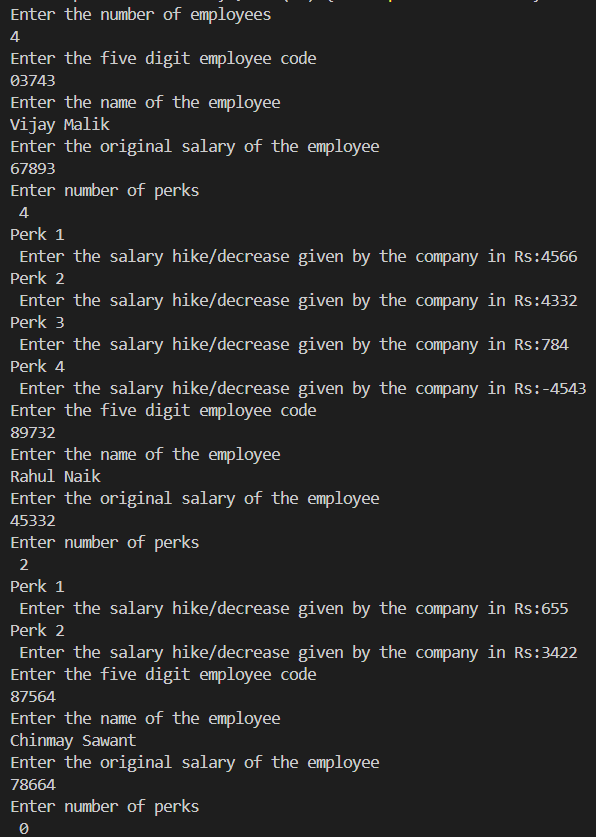
**s[i].display(state[i]) ;**

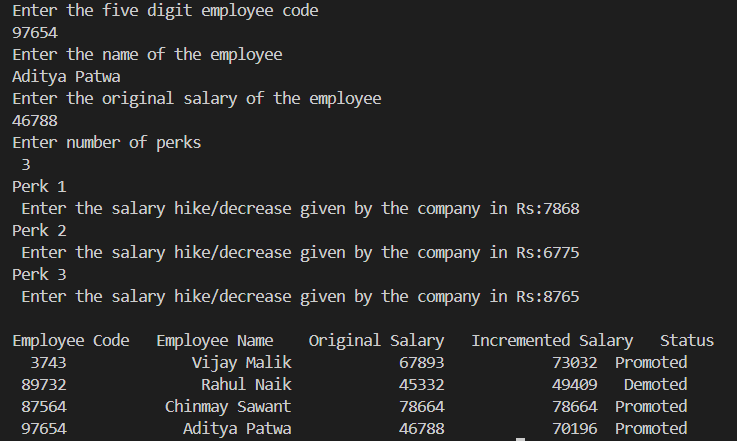
**}**

**return 0 ;**

**}**

**OUTPUT –**

****

****

**QUESTION 2**

**Implement friend function that take multiple parameters.Define a class Point with private members for x and y coordinates .Implement a friend function distance() that calculates the distance between two point objects.Create two Points objects and use the friend function to calculate and print the distance between them .**

**Program –**

**#include <iostream>**

**#include <math.h>**

**using namespace std ;**

**class Point2 ;**

**class Point1 {**

**private :**

**int x ;**

**int y ;**

**public :**

**void input(){**

**cout << "Enter the x coordinates of the first point\n" ;**

**cin>>x ;**

**cout << "Enter the y coordinates of the first point\n" ;**

**cin>>y ;**

**return ;**

**}**

**friend float distance(Point1,Point2) ;**

**};**

**class Point2 {**

**private :**

**int x ;**

**int y ;**

**public :**

**void input(){**

**cout << "Enter the x coordinates of the second point\n" ;**

**cin>>x ;**

**cout << "Enter the y coordinates of the second point\n" ;**

**cin>>y ;**

**return ;**

**}**

**friend float distance(Point1,Point2) ;**

**};**

**float distance (Point1 p1,Point2 p2){**

**int i=p1.x;**

**int j =p1.y ;**

**int k =p2.x;**

**int l =p2.y;**

**int xm = i-k ;**

**xm = xm\*xm ;**

**int ym = j-l ;**

**ym = ym\*ym ;**

**float distance = xm + ym ;**

**distance = sqrt(distance) ;**

**return distance ;**

**}**

**int main (){**

**Point1 p1 ;**

**Point2 p2 ;**

**p1.input() ;**

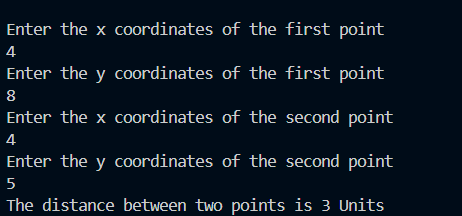
**p2.input() ;**

**cout <<"The distance between two points is "<<distance(p1,p2) <<" Units"<<endl ;**

**return 0 ;**

**}**

**OUTPUT -**

****

|  |
| --- |
| QUESTION 3  Define a class string that could work as a user defined string type.Include different types of constructors that will enable us to create an uniniatialized string as String S1;and also to initialise an object with string defined by the user like String S2.Include a function that adds two strings to make a third string .Write a complete program and also show the use of destructors to delete an object |

created

## Program –

#include <iostream>

#include <cstring>

using *namespace* std;

*class* String { *private:* *char* \*p; *int* length;

*public:* // Default constructor String() { p = new *char*[1]; p[0] = '\0'; // Initialize with an empty string length = 0;

}

// Parameterized constructor String(const *char* \**str*) { length = strlen(*str*); p = new *char*[length + 1]; // Allocate memory for the string

strcpy(p, *str*); // Copy the string

}

// Concatenation of two String objects *void* join(const String &*s1*, const String &*s2*) { length = *s1*.length + *s2*.length; p = new *char*[length + 1]; // Allocate

memory for the concatenated string

strcpy(p, *s1*.p); // Copy the first string strcat(p, *s2*.p); // Concatenate the

second string

}

// Displaying the string *void* display() { cout << "The desired string is: " << p << endl;

}

// Destructor to free allocated memory

~String() { delete[] p;

}};

*int* main() { String s1;

*char* s[50]; cout << "Enter a desired string: "; cin.getline(s, 50);

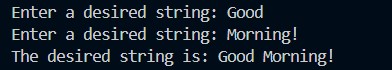
String s2(s);

cout << "Enter a desired string: "; cin.getline(s, 50); String s3(s); String s4 ; s4.join(s2, s3); s4.display();

// Destructors will be called automatically here when objects go out of scope return 0;

}

# OUTPUT –



QUESTION 4

Create a class Polar that represents the points on the plains polar coordinates (radius and angle

).Create a overloaded +operator for addition of two Polar quantities .”Adding” two points on the

plain can be accomplished by adding their X coordinates and then adding their Y coordinates .This

gives the X and Y coordinates of the “answer” .Thus you’ll need to convert two sets of polar

coordinates to rectangular coordinates ,add them ,then convert the resulting rectangular

representation back to polar .

Program –

#include <iostream>

#include <cmath>

using namespace std;

class polar ;

class Rectangular {

float x ;

float y ;

public :

Rectangular operator +(Rectangular &a){

Rectangular tmp ;

tmp.x = x+a.x ;

tmp.y=y+a.y ;

return tmp ;

}

void friend polrtorect (Rectangular &a,polar &b);

void friend recttopolar (Rectangular &a,polar &b);

};

class polar {

float radius ;

float angle ;

public :

void getvalue (){

cout <<"Enter the radius the point makes from

origin \n" ;

cin>>radius ;

cout <<"Enter angle it is inclined from the origin

\n" ;

cin>>angle;

}

void display (){

cout<<"The radius the point makes from origin

is "<<radius<<" Units"<<" and it is inclined by

"<<angle<<" radians"<<endl;

}

void friend polrtorect (Rectangular &a,polar &b);

void friend recttopolar (Rectangular &a,polar &b);

};

void recttopolar (Rectangular &a,polar &b){

float r,ang ;

r = (a.x\*a.x) + (a.y\*a.y) ;

r = sqrt(r) ;

ang = a.x/a.y ;

ang = atan(ang) ; //atan() function is used to

calculate tan inverse of a function

b.radius = r ;

b.angle = ang ;

}

void polrtorect (Rectangular &a,polar &b){

a.x = (b.radius)\*(cos(b.angle)) ;

a.y = (b.radius)\*(sin(b.angle)) ;

}

int main (){

polar P,Q,R;

Rectangular A,B,C ;

P.getvalue() ;

Q.getvalue() ;

polrtorect(A,P) ;

polrtorect(B,Q) ;

C = A+B ;

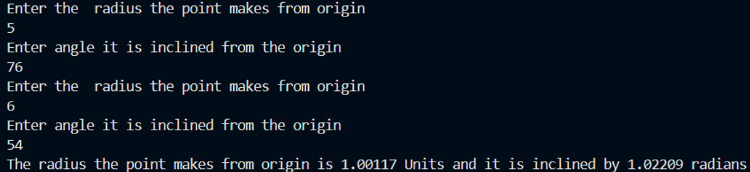
recttopolar(C,R) ;

R.display();

return 0;

}

Output –



QUESTION 5/6

**Consider an example of declaring the examination result. Design three classes: Student, Exam, and Result. The Student class has data members such as those representing roll number, name, etc. Create the class Exam by inheriting the Student class. The Exam class adds data members representing the marks scored in six subjects. Derive the Result from the Exam class and it has its own data members such as total\_marks. A new scheme for evaluation of students' performance is formulated that gives also weightage for sports. Extend the inheritance relation discussed such that the Result class also inherits properties of the Sports class. Note that the Sports class is a derived class of the Student class. Write a program to model this relationship such that members of the Student class are not inherited twice. What type of inheritance does this model belong to?"**

**PROGRAM –**

#include <iostream>

#include <iomanip>

#include <cstring>

using namespace std;

class Student {

protected:

    char \*roll\_number;

    char name[50];

public:

    Student() {

        roll\_number = nullptr;

        name[0] = '\0';

    }

    void input(char \*r, const char \*n) {

        delete[] roll\_number;

        roll\_number = new char[strlen(r) + 1];

        strcpy(roll\_number, r);

        strcpy(name, n);

    }

};

class test : virtual public Student {

protected:

    int n;

    char \*\*subject;

    int \*marks;

    int \*max\_marks;

public:

    test() {

        n = 0;

        subject = NULL;

        marks = NULL;

        max\_marks = NULL;

    }

     ~test() {

        if (subject != NULL) {

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

                delete[] subject[i];

            }

            delete[] subject;

        }

        delete[] marks;

        delete[] max\_marks;

    }

    void getinfo() {

        cout << "Enter the number of subjects: ";

        cin >> n;

        delete[] subject;

        delete[] marks;

        delete[] max\_marks;

        subject = new char\*[n];

        marks = new int[n];

        max\_marks = new int[n];

        cin.ignore();

        for (int i = 0; i < n; i++) {

            subject[i] = new char[50];

            cout << "Enter the subject name: ";

            cin.getline(subject[i], 50);

            cout << "Enter the marks obtained: ";

            cin >> marks[i];

            cout << "Enter the maximum marks: ";

            cin >> max\_marks[i];

            cin.ignore();

        }

    }

};

class sports : virtual public Student {

protected:

    string eval = "Sports";

    int score;

    int total;

public:

    sports() {

        score = 0;

        total = 0;

    }

    void getdata() {

        cout << "Enter the sports score of the student: ";

        cin >> score;

        cout << "Enter the total sports score: ";

        cin >> total;

    }

    void display() {

        cout << "The score of the student in " << eval << " is: " << score << endl;

        cout << "The total score in " << eval << " is: " << total << endl;

    }

};

class result : public test, public sports {

protected:

    int total\_marks = 0;

    int total\_max\_marks = 0;

    float percentage;

    int sports\_grade;

public:

    result() : Student() {

        total\_marks = 0;

        total\_max\_marks = 0;

        percentage = 0;

        sports\_grade = 0;

    }

    void calculateResult() {

        for (int i = 0; i < n; i++) {

            total\_marks += marks[i];

            total\_max\_marks += max\_marks[i];

        }

        total\_marks += score;

        total\_max\_marks += total;

        if (total\_max\_marks > 0) {

            percentage = ((float)total\_marks / total\_max\_marks) \* 100;

        } else {

            percentage = 0;

        }

    }

    int calc(int marks) {

        if (marks <= 100 && marks >= 85) return 10;

        else if (marks < 85 && marks >= 75) return 9;

        else if (marks < 75 && marks >= 65) return 8;

        else if (marks < 65 && marks >= 55) return 7;

        else if (marks < 55 && marks >= 50) return 6;

        else if (marks < 50 && marks >= 40) return 5;

        else return 0;

    }

    char grap(int grade) {

        if (grade == 10) return 'O';

        else if (grade == 9) return 'A';

        else if (grade == 8) return 'B';

        else if (grade == 7) return 'C';

        else if (grade == 6) return 'D';

        else if (grade == 5) return 'E';

        else return 'F';

    }

    void display() {

        cout << "\n\nRoll Number: " << roll\_number << endl;

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

        cout << "Semester RESULTS :\n";

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

        cout << " :       SUBJECT                 :   MARKS   : TOTAL   :   GRADE :\n";

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

        for (int i = 0; i < n; i++) {

            int mar = (marks[i] \* 100) / max\_marks[i];

            int grade = calc(mar);

            char grade\_letter = grap(grade);

            cout << " : " << left << setw(30) << subject[i] << " :   "

                 << fixed << setprecision(1) << marks[i] << "    :   "

                 << max\_marks[i] << "     :   "

                 << setw(2) << grade\_letter << "     :\n";

        }

        int sports\_max = (score \* 100) / total;

        char sports\_grade\_letter = grap(calc(sports\_max));

        cout << " : " << left << setw(30) << eval << " :   "

             << fixed << setprecision(1) << score << "    :   "

             << total << "     :   "<< sports\_grade\_letter << "     :\n";

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

        cout << "Percentage: " << fixed << setprecision(2) << percentage << "%" << endl;

    }

};

int main() {

//This type of inheritance is called HYBRID inheritance

    char roll[50];

    char name[50];

    cout << "Enter Roll Number: ";

    cin.getline(roll, 50);

    cout << "Enter Name: ";

    cin.getline(name, 50);

    result r;

    r.input(roll, name);

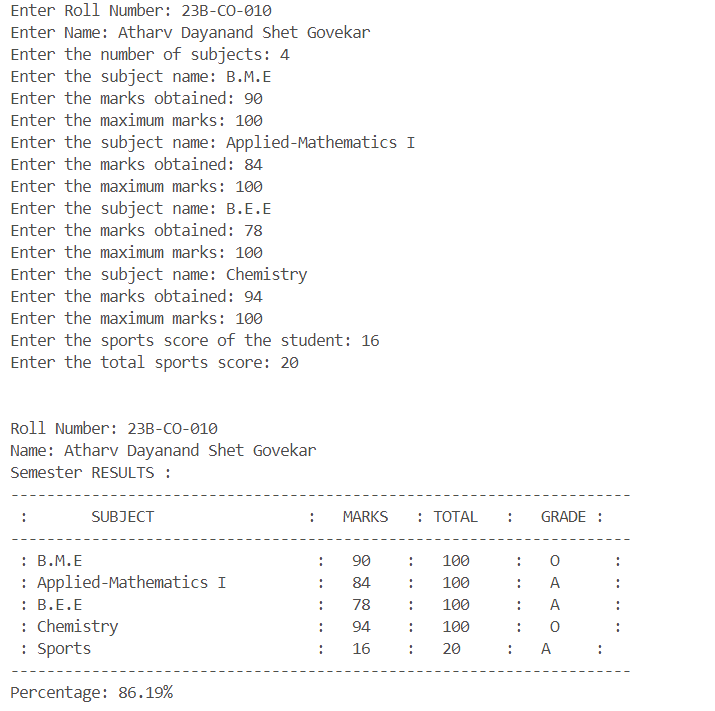
    r.getinfo();

    r.getdata();

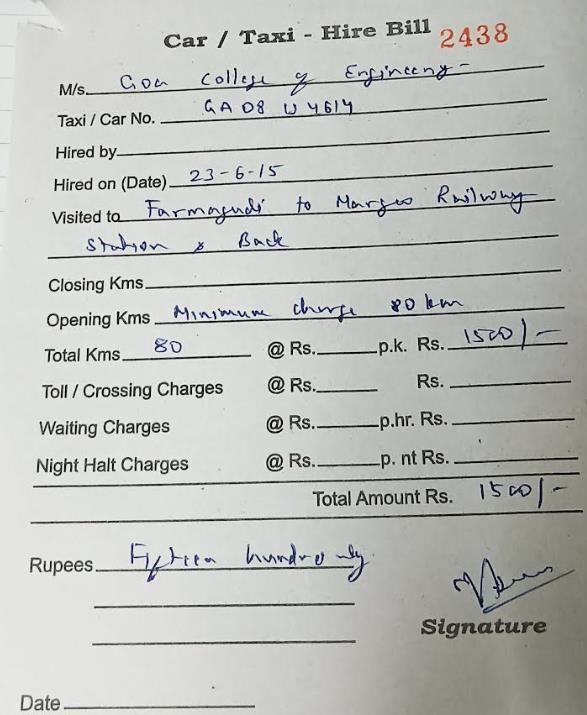
    r.calculateResult();

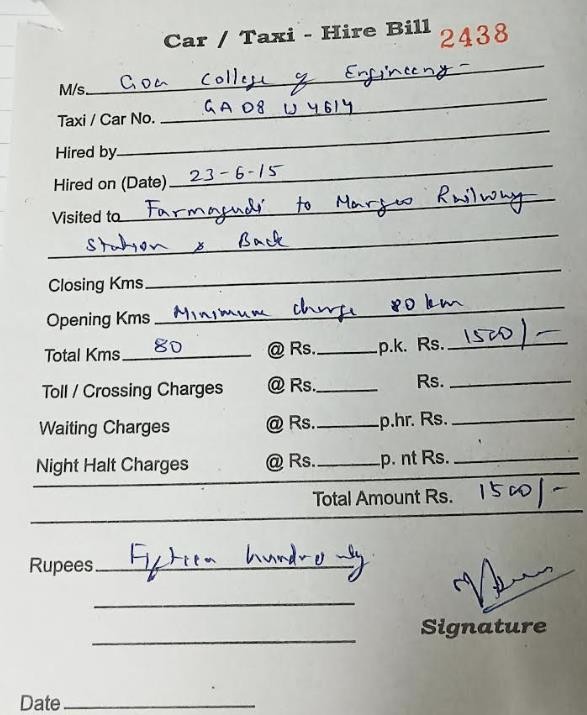
    r.display();

    return 0;}

**OUTPUT**

QUESTION 7

**[D]Write a C++ code to replicate the table given below using the concepts of Manipulators**



|  |  |
| --- | --- |
| **Program –**  **#include <iostream> #include <iomanip> #include <string> #include <vector> #include <ctime> #include <cstdlib> #include <sstream>**  **#define underline "\033[4m" using namespace std;**  **string singleDigit[] = {"", "one", "two", "three", "four",**  **"five", "six", "seven", "eight", "nine"};**  **string twoDigits[] = {"", "ten", "eleven", "twelve", "thirteen", "fourteen",**  **"fifteen", "sixteen", "seventeen", "eighteen", "nineteen"}; string tensMultiple[] = {"", "", "twenty", "thirty", "forty",**  **"fifty", "sixty", "seventy", "eighty", "ninety"};**  **string powerOfTen[] = {"", "thousand", "million", "billion"};**  ***// Function to convert numbers less than 1000 to words***  **string convertToWordsBelowThousand(int n) { string result = "";**  **if (n >= 100) {**  **result += singleDigit[n / 100] + " hundred"; n %= 100;**  **if (n) result += " and ";**  **}**  **if (n >= 20) {**  **result += tensMultiple[n / 10]; n %= 10;**  **if (n) result += " " + singleDigit[n];**  **} else if (n >= 10) {**  **result += twoDigits[n % 10];**  **} else if (n > 0) {**  **result += singleDigit[n];**  **}**  **return result;**  **}**  ***// Function to convert a number to words***  **string convertToWords(int n) { if (n == 0) return "zero";**  **string result = ""; int count = 0;**  **while (n > 0) {**  **int part = n % 1000; if (part > 0) {**  **string partWord =**  **convertToWordsBelowThousand(part);** | **Output –** |
|  |

|  |  |
| --- | --- |
| **if (count > 0 && partWord != "") partWord += " " + powerOfTen[count];**  **result = partWord + (result.empty() ? "" : " ") + result;**  **}**  **n /= 1000;**  **count++;**  **}**  **return result;**  **}**  **int generateRandomNumber() {**  **srand(time(0)); *// Seed the random number generator with the current time***  **return 1000 + rand() % 9000; *// Ensures the number is between 1000 and 9999***  **}**  **string getCurrentDateTime() { time\_t now = time(0);**  **tm \*ltm = localtime(&now);**  **ostringstream oss;**  **oss << setw(2) << setfill('0') << ltm->tm\_mday << "-"**  **<< setw(2) << setfill('0') << 1 + ltm->tm\_mon << "-"**  **<< 1900 + ltm->tm\_year << " "**  **<< setw(2) << setfill('0') << ltm->tm\_hour << ":"**  **<< setw(2) << setfill('0') << ltm->tm\_min << ":"**  **<< setw(2) << setfill('0') << ltm->tm\_sec;**  **return oss.str();**  **}**  **class Bill { public:**  **long int bill\_num = 2437; string customer;**  **int BILL\_N ;**  **string carnumber; string hiringperson; string hiring\_date; string visiting\_route;**  **string closing\_kilometer; string opening\_kilometer; int close\_kilo ;**  **float close\_charge ; int open\_kilo ; float open\_charge ; int total\_kilometer;**  **float charge\_perkm; float total\_charge; float toll\_charge;**  **float waiting\_chargeperhr; int total\_tym;**  **float Total\_wait\_charge;** |  |

cout << "Enter waiting charge per hour: "; cin >> waiting\_chargeperhr;

cout << "Enter total waiting time (in hours): "; cin >> total\_tym;

Total\_wait\_charge = (total\_tym \* waiting\_chargeperhr); cout << "Enter night charge per hour: ";

cin >> night\_chargeperhr;

cout << "Enter total nights: "; cin >> total\_nights;

Total\_night\_charge = (night\_chargeperhr \* total\_nights); total = total\_charge + toll\_charge + Total\_wait\_charge +

Total\_night\_charge;

charge\_in\_words = convertToWords(total);

time = getCurrentDateTime();

**}**

void displayDetails() {

cout << setw(104) << setfill('\_') << "" << endl;

cout << "|" << setw(102) << setfill(' ') << "" << "|" << endl;

cout << "|" << setw(38) << setfill(' ') << "" << "NARAYAN TRANSPORTS" << setw(46) << setfill(' ') << "" << "|" << endl;

cout << "|" << setw(38) << setfill(' ') << "" << "Car / Taxi - Hire Bill" << setw(42) << setfill(' ') << "" << "|" << endl;

cout << "|" << setw(64) << setfill(' ') << "" << right << bill\_num << setw(34) << setfill(' ') << "" << "|" << endl;

cout << "|" << setw(102) << setfill(' ') << "" << "|" << endl; cout << "|" << setw(15) << setfill(' ') << "" << left <<

setw(15) << setfill('\_') << "M/s "

<< underline << setw(60) << setfill('\_') << customer << "\033[0m" << setw(12) << setfill(' ') << "" << "|" << endl;

cout << "|" << setw(14) << setfill(' ') << "" << left << setw(25) << setfill('\_') << "Taxi/Car No."

<< underline << setw(50) << setfill('\_') << carnumber << "\033[0m" << setw(13) << setfill(' ') << "" << "|" << endl;

cout << "|" << setw(14) << setfill(' ') << "" << left << setw(15) << setfill('\_') << "Hired by."

<< underline << setw(60) << setfill('\_') << hiringperson

<< "\033[0m" << setw(13) << setfill(' ') << "" << "|" << endl;

cout << "|" << setw(14) << setfill(' ') << "" << left << setw(25) << setfill('\_') << "Hired on (Date)-"

<< underline << setw(50) << setfill('\_') << hiring\_date << "\033[0m" << setw(13) << setfill(' ') << "" << "|" << endl;

cout << "|" << setw(14) << setfill(' ') << "" << left << setw(15) << setfill('\_') << "Visited to"

<< underline << setw(60) << setfill('\_') << visiting\_route

<< "\033[0m" << setw(13) << setfill(' ') << "" << "|" << endl;

cout << "|" << setw(102) << setfill(' ') << "" << "|" << endl; cout << "|" << setw(14) << setfill(' ') << "" << left <<

setw(15) << setfill('\_') << "Closing Kms."

<< underline << setw(60) << setfill('\_') << closing\_kilometer << "\033[0m" << setw(13) << setfill(' ') << "" << "|" << endl;

cout << "|" << setw(14) << setfill(' ') << "" << left << setw(20) << setfill('\_') << "Opening Kms"

<< underline << setw(55) << setfill('\_') << opening\_kilometer << "\033[0m" << setw(13) << setfill(' ') << "" << "|" << endl;

cout << "|" << setw(14) << setfill(' ') << "" << left << setw(20) << setfill('\_') << "Total kms"

<< underline << setw(10) << setfill('\_') << total\_kilometer << "\033[0m" << setw(8) << setfill('\_')

<< " @ Rs." << setw(15) << setfill('\_') <<setprecision(2)

<< fixed

<< underline << charge\_perkm << "\033[0m"

<< setw(11)<< setfill(' ') << left << " p.k." << setw(2) << setfill(' ') << left << " Rs."

<< setw(9) << setfill('\_') << underline << fixed

<<setprecision(2) <<total\_charge << "\033[0m" << setw(5)

<< setfill(' ') << "" << "|" << endl;

cout << "|" << setw(14) << setfill(' ') << "" << left << setw(30) << "Toll/Crossing Charges"

<< setw(8) << setfill('\_') << " @ Rs." << setw(15)<< fixed

<<setprecision(2) << setfill('\_') << underline << toll\_charge << "\033[0m"

<< setw(4) <<setfill(' ') << left << " " << setw(5) << setfill(' ') << left << " Rs."

<< setw(15)<< fixed <<setprecision(2) << setfill('\_')

<<underline<< toll\_charge<<"\033[0m" << setw(9) << setfill(' ') << "" << "|" << endl;

cout << "|" << setw(14) << setfill(' ') << "" << left << setw(30) << "Waiting Charges"

<< setw(7) << setfill('\_') << " @ Rs." << setw(13) << fixed

<<setprecision(2)<< setfill('\_') << underline << waiting\_chargeperhr << "\033[0m"

<< setw(10) <<setfill(' ')<< left << " p.h.r " << setw(3) << setfill('\_') << left << " Rs."

<< setw(15) << fixed <<setprecision(2) << setfill('\_')

<<underline<< Total\_wait\_charge <<"\033[0m" << setw(5)

<< setfill(' ') << "" << "|" << endl;

cout << "|" << setw(14) << setfill(' ') << "" << left << setw(30) << "Night Halt Charges"

<< setw(7) << setfill('\_') << " @ Rs." << setw(15)<< fixed

<<setprecision(2) << setfill('\_') << underline << night\_chargeperhr << "\033[0m"

<< setw(5) << left << " p.nt " << setw(5) << setfill('\_') << left << " Rs."

<< setw(15)<< fixed <<setprecision(2) << setfill('\_')

<<underline<< Total\_night\_charge <<"\033[0m"<< setw(5)

<< setfill(' ') << "" << "|" << endl;

cout << "|" << setw(102) << setfill(' ') << "" << "|" << endl; cout << "|" << setw(14) << setfill(' ') << "" << setw(75) <<

setfill('\_') << ""

<< setw(13) << setfill(' ') << "" << "|" << endl;

cout << "|" << setw(14) << setfill(' ') << "" << setw(50) << setfill(' ')

<< "" << "Total -> " << setw(19) << total << setw(10) << setfill(' ') << "" << "|" << endl;

cout << "|" << setw(14) << setfill(' ') << "" << setw(75) << setfill('\_') << ""

<< setw(13) << setfill(' ') << "" << "|" << endl;

cout << "|" << setw(102) << setfill(' ') << "" << "|" << endl; cout << "|" << setw(14) << setfill(' ') << "" << left << setw(6)

<< "Rupees - " << underline << setw(59) << setfill(' ') << left << charge\_in\_words << "\033[0m" << setw(12)

<< setfill(' ') << "" << setw(6) << setfill(' ') << " (T.B) " << setw(1) << setfill(' ') << "" << "|" << endl;

cout << "|" << setw(24) << setfill(' ') << "" << "Time: " << underline << time << "\033[0m" << setw(15)

<< setfill(' ') << "" << setw(37) <<right<< setfill(' ') << " SIGNATURE " << setw(1) << setfill(' ') << "" << "|" << endl;

cout << "|" << setw(102) << setfill(' ') << "" << "|" << endl;

cout << "|" << setw(102) << setfill('\_') << "" << "|" << endl;

**}**

**};**

int main() {

***// #ifndef ATHARV\_CUSTOM***

***// freopen("in.txt","r",stdin);***

***// #endif***

**}**

**return 0;**

**Bill bill;**

**char choice; int n ;**

**cout<<"Enter the number of people you want to make the bill for\n"<<endl ;**

**cin>>n ;**

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

**cout <<"====================="<<"Constumer "<<i+1**

**<<"=========================";**

**cout <<endl ; do {**

**cout << "\n\n===== Car/Taxi Hire Bill Menu =====\n"; cout << "1. Input Bill Details\n";**

**cout << "2. Display Bill Details\n"; cout << "3. Exit\n";**

**cout << "Enter your choice: "; cin >> choice;**

**cin.ignore();**

**switch (choice) { case '1':**

**bill.inputDetails(); break;**

**case '2':**

**bill.displayDetails(); break;**

**case '3':**

**cout << "Exiting the program. \n \n \n\n"; break;**

**default:**

**cout << "Invalid choice. Please try again.\n"; break;**

**}**

**} while (choice != '3');**

**}**

Question 8 - Write a C++ program to a budgeting application that allows user to log income and expense . Store transactions in a CSV file and generate monthly reports.

|  |  |
| --- | --- |
| Program –  #include <iostream>  #include <fstream>  #include <iomanip>  #include <string>  using namespace std;  *// Function prototypes with definitions*  void displayMenu() {      cout << "\n------ Budgeting Application ------\n";      cout << "1. Log a Transaction\n";      cout << "2. Generate Monthly Report\n";      cout << "3. View All Transactions\n";      cout << "4. Delete All Transactions\n";      cout << "5. Exit\n";  }  void logTransaction() {      ofstream file("transactions.csv", ios::app);      if (!file) {          cerr << "Error opening file." << endl;          return;      }      string type, description;      double amount;      cout << "Enter type (income/expense): ";      cin >> type;      cout << "Enter description: ";      cin.ignore();      getline(cin, description);      cout << "Enter amount: ";      cin >> amount;      file << type << "," << description << "," << amount << "\n";      file.close();      cout << "Transaction logged successfully." << endl;  }  void generateMonthlyReport() {      ifstream file("transactions.csv");      if (!file) {          cerr << "Error opening file." << endl;          return;      }      string line, type, description;      double amount, totalIncome = 0, totalExpense = 0;      while (getline(file, line)) {          size\_t pos1 = line.find(',');          size\_t pos2 = line.rfind(',');            if (pos1 != string::npos && pos2 != string::npos && pos1 != pos2) {              type = line.substr(0, pos1);              description = line.substr(pos1 + 1, pos2 - pos1 - 1);              amount = stod(line.substr(pos2 + 1));              if (type == "income") {                  totalIncome += amount;              } else if (type == "expense") {                  totalExpense += amount;              }          }      }      file.close();      cout << fixed << setprecision(2);      cout << "\n------ Monthly Report ------\n";      cout << "Total Income: Rs" << totalIncome << endl;      cout << "Total Expenses: Rs" << totalExpense << endl;      cout << "Net Savings: Rs" << (totalIncome - totalExpense) << endl;  }  void viewAllTransactions() {      ifstream file("transactions.csv");      if (!file) {          cerr << "Error opening file." << endl;          return;      }      string line, type, description;      double amount;      cout << "\n------ All Transactions ------\n";      cout << left << setw(15) << "Type" << setw(25) << "Description" << "Amount" << endl;      cout << "-----------------------------------------------\n";      while (getline(file, line)) {          size\_t pos1 = line.find(',');          size\_t pos2 = line.rfind(',');          if (pos1 != string::npos && pos2 != string::npos && pos1 != pos2) {              type = line.substr(0, pos1);              description = line.substr(pos1 + 1, pos2 - pos1 - 1);              amount = stod(line.substr(pos2 + 1));              cout << left << setw(15) << type << setw(25) << description << fixed << setprecision(2) << amount << endl;          }      }      file.close() | OUTPUT –      }  void deleteAllTransactions() {  ofstream file("transactions.csv", ios::trunc);  if (!file) {  cerr << "Error opening file." << endl;  return;  }  file.close();  cout << "All transactions have been deleted." << endl;  }  int main() {  int choice;  while (true) {  displayMenu();  cout << "Enter your choice: ";  cin >> choice;  switch (choice) {  case 1:  logTransaction();  break;  case 2:  generateMonthlyReport();    break;  case 3:  viewAllTransactions();  break;  case 4:  deleteAllTransactions();  break;  case 5:  cout << "Exiting the program." << endl;  return 0;  default:  cout << "Invalid choice. Please try again." << endl;  }  }  return 0;  } |

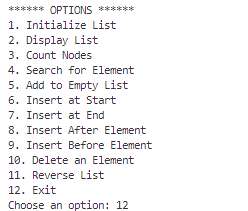
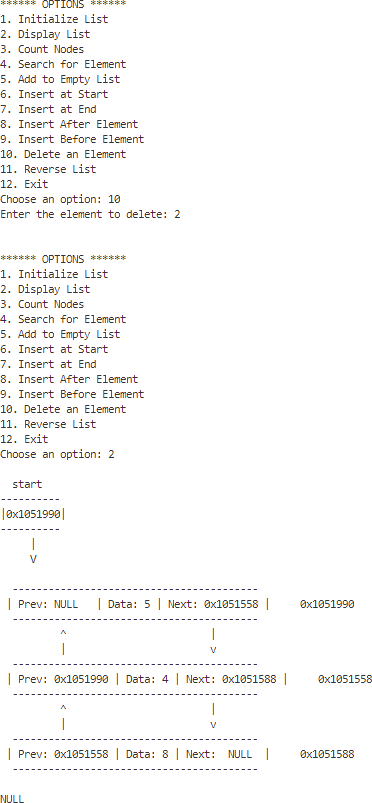
Question 9 –

# Write an interactive program for creating a doubly linked list .The program must support insertion and deletion of a node ,The doubly Linked List class must be of template type

|  |  |
| --- | --- |
| **Program –**  **#include <iostream> using namespace std;**  **template <typename T> struct Node {**  **T value;**  **Node<T>\* previous; Node<T>\* next;**  **};**  **template <typename T> class DoublyLinkedList { public:**  **DoublyLinkedList() : start(nullptr) {}**  **void initList(T data) {**  **start = new Node<T>(); start->value = data;**  **start->previous = nullptr; start->next = nullptr;**  **}**  **void insertAtBeginning(T data) { Node<T>\* newNode = new Node<T>(); newNode->value = data;**  **newNode->previous = nullptr; newNode->next = start;**  **if (start) {**  **start->previous = newNode;**  **}**  **start = newNode;**  **}**  **void insertAtEnd(T data) {**  **Node<T>\* newNode = new Node<T>(); Node<T>\* tempNode = start;**  **while (tempNode->next != nullptr) { tempNode = tempNode->next;**  **}**  **newNode->value = data; newNode->previous = tempNode; newNode->next = nullptr; tempNode->next = newNode;**  **}**  **void insertAfter(T data, T item) { Node<T>\* newNode = new Node<T>(); Node<T>\* tempNode = start;**  **while (tempNode != nullptr) {**  **if (tempNode->value == item) { newNode->value = data; newNode->previous = tempNode;**  **newNode->next = tempNode->next;** | **Output –** |
|  |

|  |  |
| --- | --- |
| **if (tempNode->next != nullptr) { tempNode->next->previous = newNode;**  **}**  **tempNode->next = newNode; return;**  **}**  **tempNode = tempNode->next;**  **}**  **cout << "Item not found\n";**  **}**  **void insertBefore(T item, T data) { Node<T>\* newNode = new Node<T>(); Node<T>\* tempNode = start;**  **if (start == nullptr) {**  **cout << "\nThe list is empty\n"; return;**  **}**  **if (start->value == item) { newNode->value = data; newNode->next = start; start = newNode;**  **return;**  **}**  **while (tempNode->next != nullptr) {**  **if (tempNode->next->value == item) { newNode->value = data;**  **newNode->next = tempNode->next; tempNode->next = newNode;**  **return;**  **}**  **tempNode = tempNode->next;**  **}**  **cout << "\nItem not found\n";**  **}**  **void createList() { int numElements; T data;**  **cout << "How many elements would you like to add? "; cin >> numElements;**  **if (numElements == 0) { return;**  **}**  **cout << "Enter the first element: "; cin >> data;**  **initList(data);**  **for (int i = 1; i < numElements; i++) { cout << "Enter the next element: "; cin >> data;**  **insertAtEnd(data);**  **}**  **}**  **void printList() const { Node<T>\* temp = start;**  **// Print start pointer in a box pointing to the first node cout << "\n start\n";**  **cout << " \n";** |  |

cout << "|" << start << "|\n"; cout << " \n";



cout << " | \n"; if (start != nullptr) {

cout << " V \n\n";

}

while (temp != nullptr) {

if (temp->previous == nullptr) {

cout << " \n";

cout << " | Prev: NULL | Data: " << temp->value << " | Next: " << temp->next << " | " << temp << " \n";

cout << " \n";

} else if (temp->next == nullptr) {

cout << " \n";

cout << " | Prev: " << temp->previous << " | Data: "

<< temp->value << " | Next: NULL | " << temp << " \n"; cout << " \n";

} else {

cout << " \n";

cout << " | Prev: " << temp->previous << " | Data: "

<< temp->value << " | Next: " << temp->next << " | " << temp << " \n";

cout << " \n";

}

if (temp->next != nullptr) {

cout << " ^ |\n";

cout << " | v\n";

}

temp = temp->next;

}

cout << "\nNULL\n";

}

int countNodes() const { int count = 0;

for (Node<T>\* tempNode = start; tempNode != nullptr; tempNode = tempNode->next) {

count++;

}

return count;

}

Node<T>\* search(T item) const { Node<T>\* tempNode = start; while (tempNode != nullptr) {

if (tempNode->value == item) { return tempNode;

}

tempNode = tempNode->next;

}

return nullptr;

}

void removeNode(T data) { Node<T>\* tempNode = start; Node<T>\* prevNode = nullptr; while (tempNode != nullptr) {

if (tempNode->value == data) {

if (prevNode != nullptr) {

prevNode->next = tempNode->next;

} else {

start = tempNode->next;

}

if (tempNode->next != nullptr) { tempNode->next->previous = prevNode;

}

delete tempNode; return;

}

prevNode = tempNode; tempNode = tempNode->next;

}

cout << "Element not found\n";

}

void reverseList() {

Node<T>\* tempNode = nullptr; Node<T>\* current = start; while (current != nullptr) {

tempNode = current->previous; current->previous = current->next; current->next = tempNode; current = current->previous;

}

if (tempNode != nullptr) {

start = tempNode->previous;

}

}

void swapAlternateNodes() {

if (start == nullptr || start->next == nullptr) { return;

}

Node<T>\* tempNode1 = start; Node<T>\* tempNode2 = start->next; start = tempNode2;

Node<T>\* temp;

while (true) {

temp = tempNode2->next; tempNode2->next = tempNode1;

tempNode2->previous = tempNode1->previous; tempNode1->previous = tempNode2; tempNode1->next = temp;

if (temp == nullptr || temp->next == nullptr) { break;

}

tempNode1->next = temp->next; tempNode1 = temp;

tempNode2 = tempNode1->next;

}

}

private: Node<T>\* start;

};

**}**

**cin >> item;**

**list.insertAfter(data, item); break;**

**case 9:**

**cout << "Enter the value to add: "; cin >> data;**

**cout << "Enter the element to insert before: "; cin >> item;**

**list.insertBefore(item, data); break;**

**case 10:**

**cout << "Enter the element to delete: "; cin >> data;**

**list.removeNode(data); break;**

**case 11:**

**list.reverseList();**

**cout << "List reversed\n"; break;**

**case 12 :**

**break; default:**

**cout << "Invalid option\n"; break;**

**}**

**} while (option != 12);**

**return 0;**

Question 10 – Write a C++ program using string library and inbuilt functions to find and replace patterns in the string .Also implement validation of email addeesses and phone numbers

|  |  |
| --- | --- |
| **Program –**  #include <iostream>  #include <cstring>    using namespace std;    *// Function to replace a pattern in a string* void *replacePattern*(string &s, const string &pattern, const string &replacement) {  int pos = s.*find*(pattern); while (pos != -1) {  s.*replace*(pos, pattern.*length*(),  replacement);  pos = s.*find*(pattern, pos +  replacement.*length*());  }  }    *// Function to validate an email address* bool *isValidEmail*(const string &email) {  int atPos = email.*find*('@'); int dotPos = email.*find*('.');  return (atPos != -1 && dotPos != -1 && atPos < dotPos);  }    *// Function to validate a phone number*  bool *isValidPhoneNumber*(const string &phone)  {  if (phone.*length*() != 10) {  return false;  }  for (int i = 0; i < phone.*length*(); i++) {  if (!*isdigit*(phone*[*i*]*)) { return false;  }  }  return true;  } | **Output –** |

|  |  |
| --- | --- |
| int *main*() { int choice;  do {  cout *<<* "Menu:\n";  cout *<<* "1. Replace pattern in string\n"; cout *<<* "2. Validate email address\n"; cout *<<* "3. Validate phone number\n"; cout *<<* "4. Exit\n";  cout *<<* "Enter your choice: ";  cin *>>* choice;    switch (choice) {  case 1: {  string s, pattern, replacement; cout *<<* "Enter a string: ";  cin.*ignore*(); *// To ignore the newline*  *character left by previous input*  *getline*(cin, s);  cout *<<* "Enter the pattern you want to replace: ";  cin *>>* pattern; cout *<<* "Enter the replacement string:  ";  cin *>>* replacement;  *replacePattern*(s, pattern, replacement);  cout *<<* "Modified string: " *<<* s *<<* *endl*; break;  } case 2: { string email;  cout *<<* "Enter an email address to validate: "; | cin *>>* email;  if (*isValidEmail*(email)) {  cout *<<* "Valid email address." *<<* *endl*;  } else {  cout *<<* "Invalid email address." *<<* *endl*;  }  break;  } case 3: {  string phone;    cout *<<* "Enter a phone number to validate: "; cin *>>* phone;  if (*isValidPhoneNumber*(phone)) {  cout *<<* "Valid phone number." *<<* *endl*;  } else {  cout *<<* "Invalid phone number." *<<* *endl*;  }  break;  } case 4: cout *<<* "Exiting..." *<<* *endl*; break; default:  cout *<<* "Invalid choice. Please try again." *<<* *endl*;  }  } while (choice != 4);    return 0;  } |

Question 11 - Write a C++ program to implement a queue class that supports various data types and allows users to insert and remove items. The queue should maintain two and handle exceptions for exceeding capacity and removing from an empty queue. Include an interactive main() function that lets users test the queue functionality and recover from exceptions without corrupting the queue's contents.

|  |  |
| --- | --- |
| **Program –**  **#include <iostream>**  **#include <stdexcept>**    **using namespace std;**    ***// Template class for Queue* template <typename T>**  **class Queue { private:**  **static const int capacity = 10; *// Fixed size for simplicity* T data[capacity]; *// Array to store queue elements***  **int front; *// Index of the front of the queue* int rear; *// Index of the rear of the queue* int count; *// Current number of elements in the queue***    **public:**  ***// Constructor to initialize the queue***  ***Queue*() : *front*(0), *rear*(0), *count*(0) {} *// Simple constructor***    ***// Enqueue operation* void *enqueue*(T item) {**  ***// Check if the queue is full* if (count >= capacity) {**  ***// Throw an exception if capacity is exceeded* throw *overflow\_error*("Queue capacity exceeded!");**  **}**  ***// Insert the item at the rear* data[rear] = item;**  ***// Update the rear index, wrapping around if necessary* rear = (rear + 1) % capacity;  *// Increment the count of items* count++;**  **}**    ***// Dequeue operation***  **T *dequeue*() {**  ***// Check if the queue is empty***  **if (count <= 0) {**  ***// Throw an exception if trying to dequeue from an empty queue***  **throw *underflow\_error*("Queue is empty!");**  **}**  ***// Retrieve the item from the front***  **T item = data[front];**  ***// Update the front index, wrapping around if necessary***  **front = (front + 1) % capacity;  *// Decrement the count of items* count--;** | **Output –** |
| ***// Return the dequeued item***  **return item;**  **}**    ***// Get the number of items in the queue* int *size*() const {**  **return count; *// Return the current count of items***  **}**    ***// Check if the queue is empty* bool *isEmpty*() const {**  **return count == 0; *// Return true if count is zero***  **}**    ***// Function to get the maximum capacity of the queue* int *getCapacity*() const {**  **return capacity; *// Return the maximum capacity***  **}**    ***// Display the elements in the queue***  **void *display*() const { if (*isEmpty*()) {**  **cout *<<* "Queue is empty.\n"; return;**  **}**  **cout *<<* "Queue elements: "; for (int i = 0; i < count; ++i) {**  **cout << data[(front + i) % capacity] << " ";**  **}**  **cout *<<* "\n";**  **}**  **};**    ***// Interactive main function***  **int *main*() {**  **int n; *// Number of values to enqueue***  **Queue<int> queue; *// Create a queue for integers* int choice, value; *// Variables for user choice and value input***    **do {**  ***// Menu for user choices* cout *<<* "\n1. Enqueue\n2. Dequeue\n3. Display\n4. Exit\nChoose an option: "; cin *>>* choice;** | try {  switch (choice) { case 1: *// Enqueue*  cout *<<* "Enter number of values to enqueue  (max " *<<* queue.*getCapacity*() *<<* "): "; cin *>>* n;  if (n > queue.*getCapacity*()) {  throw *overflow\_error*("Cannot enqueue more than queue capacity.");  }  for (int i = 0; i < n; i++) {  cout *<<* "Enter value to enqueue: "; cin *>>* value; *// Use 'value' for input* queue.*enqueue*(value); *// Call enqueue method*  }  queue.*display*(); *// Display the queue* break;    case 2: *// Dequeue*  value = queue.*dequeue*(); *// Call dequeue method*  cout *<<* value *<<* " dequeued.\n"; queue.*display*(); *// Display the queue after dequeue*  break;    case 3: *// Display*  queue.*display*(); *// Display current queue* break;    case 4: *// Exit*  cout *<<* "Exiting...\n"; break;  default:  cout *<<* "Invalid option. Please choose again.\n";  }  } catch (const overflow\_error& e) {  *// Catch overflow error if the queue is full*  cerr *<<* e.*what*() *<<* "\n"; *// Output error message*  } catch (const underflow\_error& e) {  *// Catch underflow error if the queue is empty* cerr *<<* e.*what*() *<<* "\n"; *// Output error message*  }  } while (choice != 4); *// Continue until user chooses to exit*    return 0; *// End of the program*  } |

Question 12 - Write a C++ program to implement standard library list sequence container and do bidirectional iteration and sorting

|  |  |
| --- | --- |
| **Program –**  #include <iostream>  #include <list> *// list datatype*  #include <algorithm>    using namespace std;  *//itarators are used to iterate(list, vector, map, set, etc) through the elements of the container*    void *iterateForward*(const list<int>& lst) { cout *<<* "Forward iteration: "; for (const int& val : lst) { cout *<<* val *<<* " ";  } *// for each loop is used to iterate through the elements of the list*  cout *<<* *endl*;  }    void *iterateBackward*(const list<int>& lst) { cout *<<* "Backward iteration: "; for (auto it = lst.*rbegin*(); it *!=* lst.*rend*(); *++*it) { *// rbegin() and rend() are used to iterate in reverse order*  cout *<<* *\**it *<<* " "; *// \*it is used to access the value at the iterator*  }  cout *<<* *endl*;  }    void *sortList*(list<int>& lst) {  lst.*sort*(); *// .sort() function sorts the list*  *in ascending order*  }    void *displayMenu*() { cout *<<* "Menu:" *<<* *endl*; cout *<<* "1. Display list forward" *<<* *endl*; cout *<<* "2. Display list backward" *<<* *endl*;  cout *<<* "3. Sort list" *<<* *endl*; cout *<<* "4. Exit" *<<* *endl*; cout *<<* "Enter your choice: ";  }    int *main*() {  list<int> myList; *// list datatype* int choice, value;    cout *<<* "Enter elements of the list (enter  -1 to stop): ";  while (cin *>>* value && value != -1) { myList.*push\_back*(value); *// .push\_back*  *does the role of enqueue by adding*  *elements to the end of the list*  }    do {  *displayMenu*();  cin *>>* choice;    switch (choice) { case 1:  *iterateForward*(myList); break; case 2:  *iterateBackward*(myList); break; case 3: *sortList*(myList);  cout *<<* "List sorted." *<<* *endl*; break; case 4: cout *<<* "Exiting..." *<<* *endl*; break; default:  cout *<<* "Invalid choice. Please try again." *<<* *endl*;  }  } while (choice != 4);    return 0;  } | **Output –** |