**Q1. An electricity board charges the following rates to user. For the first 100 units → 60p per unit. For the next 200 units→80p per unit. Beyond 300 units→90p per unit. All users are charged a minimum of Rs. 50; if the total amount is more than 300 then an additional surcharges of 15% is added. Write a program to accept name of user consumed and print charges with their rates**

**<Source code>**

#include<conio.h>

#include<#include<iostream.h>

string.h>

class calcharge

{

protected:

float charge,surfcharges;

int unit;

char name[20];

public:

calcharge(char nm[], int value)

{

unit=value;

strcpy(name,nm);

charge=0;

surfcharges=0;

}

void basecharge()

{

if(unit<=100)

{

charge = unit\*0.60;

}

else if(unit>100 && unit<=300)

{

charge = (100\*0.60)+(unit-100)\*0.80;

}

else

{

charge = (100\*0.60)+(200\*0.80)+(unit-300)\*0.90;

}

}

void checkmin()

{

if(charge<=50)

charge = 50;

}

void surfcharge()

{

if(charge>300)

surfcharges = charge + charge\*0.15;

}

void display()

{

cout<<"\n===========================";

cout<<"\nName: "<<name;

if(surfcharges>0)

{ cout<<"\n===========================";

cout<<"\nBase charge: "<<charge;

cout<<"\n+ surfaces charge: "<<surfcharges;

}

cout<<"\n===========================";

cout<<"\nTotal charge: "<<charge+surfcharges;

cout<<"\n===========================";

}

};

void main()

{

char nm[20];

int unit;

clrscr();

cout<<"Enter your Name: ";

cin>>nm;

cout<<"Enter your unit: ";

cin>>unit;

calcharge bill(nm, unit);

bill.basecharge();

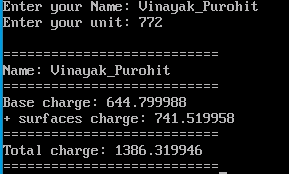
bill.checkmin();

bill.surfcharge();

bill.display();

getch();

}



**Q2. Define a class to represent a bank account. Include the following members: a. Name of the depositor b. Account number c. Type of Account d. Balance amount in the Account Member Functions: a. To assign initial values. b. To deposit an amount. c. To withdraw an amount after checking the balance. d. To display name and balance. Write main program and handle accounts of 5 customer**

**<source code>**

#include<iostream.h>

#include<conio.h>

#include<string.h>

class Bank

{

private:

long accno;

float balance;

char holdernm[15],acctype[10];

public:

void Addinfo(long accnum, char custnm[], char accnm[], float amount)

{

accno = accnum;

strcpy(holdernm,custnm);

strcpy(acctype,accnm);

balance = amount;

}

void Deposit(float amount)

{

if(amount>0)

{

balance += amount;

cout<<"\nRupees "<<amount<<" successfully added to your account";

}

else

cout<<"\nPlease enter an valid Amount";

}

void Withdraw(float amount)

{

if(amount>0)

{

if(balance>amount)

{

balance -= amount;

cout<<"\nRupees "<<amount<<" successfully deducted from your account";

}

else

cout<<"\nyou don't have sufficient balance to withwraw money";

}

else

cout<<"\nPlease Enter valid amount";

}

void Display()

{

cout<<"\nACCOUNT NO: "<<accno;

cout<<"\nHOLDER NAME:"<<holdernm;

cout<<"\nACCOUNT TYPE: "<<acctype;

cout<<"\nTOTAL BALANCE: "<<balance;

}

};

void main()

{

long accno;

float balance;

char holdernm[15],acctype[10];

Bank b[5];

clrscr();

b[1].Addinfo(123123123,"vishal","saving",18000.01);

b[2].Addinfo(99295,"Rakesh","current",35000.23);

b[3].Addinfo(23541,"mahesh","saving",12030.6);

b[4].Addinfo(98437,"shailesh","saving",9002.28);

b[5].Addinfo(93625,"chirag","current",19002.35);

// perfoming task

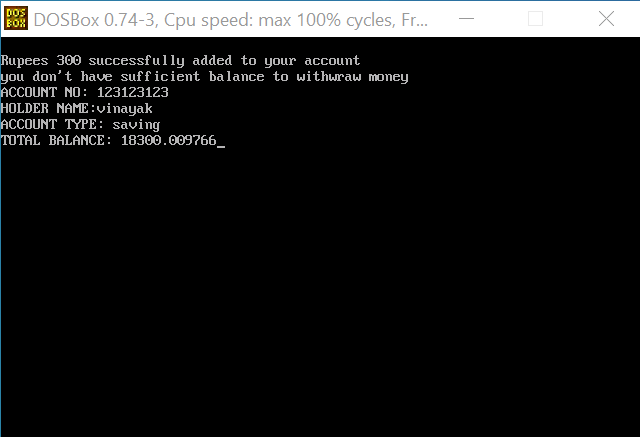
b[1].Deposit(300);

b[1].Withdraw(20000);

b[1].Display();

getch();

}



**Q3. Program to create a class person having members name and age. Derive a class student having member percentage. Derive another class teacher having member salary. Write necessary member function to initialize, read and write data. Also write the main function.**

**<source code>**

#include<iostream.h>

#include<conio.h>

class Person

{

protected:

char name[15];

int age;

public:

void getdetail()

{

cout<<"Enter Name: ";

cin>>name;

cout<<"Enter Age: ";

cin>>age;

}

};

class Student: public Person

{

protected:

float per;

public:

void getdata()

{ Person::getdetail();

cout<<"Enter percentage: ";

cin>>per;

}

void display()

{

cout<<"\nName: "<<name;

cout<<"\nAge: "<<age;

cout<<"\nPer: "<<per;

}

};

class Teacher: public Person

{

protected:

float salary;

public:

void getdata()

{

Person::getdetail();

cout<<"\nEnter salary: ";

cin>>salary;

}

void display()

{

cout<<"\nName: "<<name;

cout<<"\nAge: "<<age;

cout<<"\nSalary: "<<salary;

}

};

void main()

{

int ch;

Student s;

Teacher t;

clrscr();

cout<<"!! Select On which entity's you want to perform your Operation !!";

cout<<"\n1.Student";

cout<<"\n2.Teacher";

cin>>ch;

if(ch == 1)

{

s.getdata();

s.display();

}else if(ch == 2){

t.getdata();

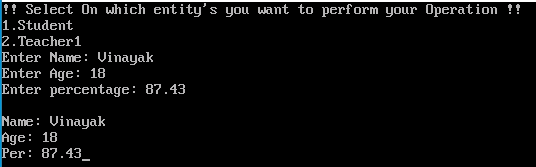
t.display();

}else

cout<<"!! Please select valid choice !!";

getch();

}



**Q4. Program to create a class name student having date member name, no & three marks. Write a member function to input name, roll no & marks & calculate percentage**

**<source code>**

#include<iostream.h>

#include<conio.h>

class Student

{

protected:

int roll\_no,marks[3];

char name[15];

float per;

public:

void getdata()

{

cout<<"Enter Rollno: ";

cin>>roll\_no;

cout<<"Enter Name: ";

cin>>name;

for(int i=0;i<3;i++)

{

cout<<"Enter marks of sub "<<i+1<<" :";

cin>>marks[i];

}

}

void calculateper()

{

int totalmarks=0;

for(int i=0;i<3;i++)

totalmarks += marks[i];

per = totalmarks/3;

}

void display()

{

cout<<"\nRoll No: "<<roll\_no;

cout<<"\nName: "<<name;

for(int i=0;i<3;i++)

cout<<"\nMarks of sub "<<i+1<<" :"<<marks[i];

cout<<"\nPer: "<<per;

}

};

void main()

{ Student s;

clrscr();

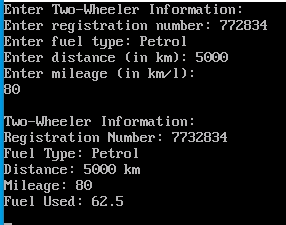
s.getdata();

s.calculateper();

s.display();

getch();

}



**5. Create a class called "Vehicle" which contains data members registration number and fuel type Make getdata() function to input data value. Create class "two-Wheeler" from vehicle which contains data member’s distance and mileage Make getdata() function to input data. Use overloading techniques for getdata() function and display the information with fuel used.**

**<source code>**

#include <iostream.h>

#include <conio.h>

class vehicle

{

protected:

int reg\_no;

char fueltype[8];

public:

void getdata() {

cout << "Enter registration number: ";

cin >> reg\_no;

cout << "Enter fuel type: ";

cin >> fueltype;

}

void display() {

cout << "Registration Number: " << reg\_no << endl;

cout << "Fuel Type: " << fueltype << endl;

}

};

class twowheeler : public vehicle

{

private:

double distance;

double mileage;

public:

void getdata() {

vehicle::getdata();

cout << "Enter distance (in km): ";

cin >> distance;

cout << "Enter mileage (in km/l): ";

cin >> mileage;

}

void display() {

vehicle :: display();

cout << "Distance: " << distance << " km" << endl;

cout << "Mileage: " << mileage << endl;

double fuelused = distance / mileage;

cout << "Fuel Used: " << fuelused <<endl;

}

};

void main() {

clrscr();

twowheeler tw;

cout << "Enter Two-Wheeler Information:" << endl;

tw.getdata();

cout << "\nTwo-Wheeler Information:" << endl;

tw.display();

getch();

}

**Q6. Write a program that consist of two classes Time12 and Time24. The first one maintains time on 12 hour basis, whereas the other one maintains it on 24-hour basis**

**<source code>**

#include <iostream.h>

#include <conio.h>

class Time12;

class Time24

{

int h, m;

public:

void getData()

{

cout << "\nEnter hour and minute (24-hour) : ";

cin >> h >> m;

}

int geth()

{

return h;

}

int getm()

{

return m;

}

void putData()

{

cout << "\n";

if (h <= 9)

cout << "0" << h << ":";

else

cout << h << ":";

if (m <= 9)

cout << "0" << m;

else

cout << m;

}

};

class Time12

{

int h, m;

public:

Time12(Time24 t)

{

h = t.geth();

if (h > 12)

{

h = h % 12;

m = t.getm();

}

}

void getData()

{

cout << "\nEnter hour and minute (12-hour) : ";

cin >> h >> m;

}

void putData()

{

cout << "\n";

if (h <= 9)

cout << "0" << h << ":";

else

cout << h << ":";

if (m <= 9)

cout << "0" << m;

else

cout << m;

}

};

void main()

{

clrscr();

Time12 t12;

Time24 t24;

t24.getData();

t12 = t24;

t12.putData();

t12.getData();

t24.putData();

getch();

}

**Q7).Create two classes DM and DB which store the values of distance. DM stores distance in meters and centimeters. DB stores distances in feet and inches. Write a program that can read values for the class object and add one object of DM with another object of DB. Use a friend function to carry out the addition operation and this function will display answer in meter and centimeters**

**<source code>**

#include <iostream.h>

#include <conio.h>

#include <string.h>

class tel\_dir {

char name[20];

int number;

public:

void ADD() {

cout << "\nEnter Name, Telephone Number:";

cin >> name >> number;

}

void show() {

cout << name << "\t" << number << endl;

}

int isExists(char \*nm)

{

return strcmp(nm, name) == 0;

}

};

int main()

{

int i;

tel\_dir entry[3];

clrscr();

cout << "\nEnter details for 3 entries\n";

for (i = 0; i < 3; i++) {

entry[i].ADD();

}

cout << "\nOUTPUT\n\n";

for (i = 0; i < 3; i++) {

entry[i].show();

}

char nm[20];

cout << "\nEnter name to search : ";

cin >> nm;

for (i = 0; i < 3; i++) {

if (entry[i].isExists(nm))

entry[i].show();

}

getch();

return 0;

}

**Q8).Write a program to maintain a telephone directory use add() and Show() methods to add new entries and display the telephone numbers of a person when the name of the person is given**

**<source code>**

#include <iostream.h>

#include<conio.h>

#include <string.h>

class tel\_dir

{

int totalrecord;

int sno;

struct Record

{

char name[20];

long number;

} records[50];

public:

tel\_dir()

{

totalrecord = 50;

sno=0;

}

void add()

{

if (sno < totalrecord)

{

cout << "\nEnter Name: ";

cin >> records[sno].name;

cout << "\nEnter Telephone Number: ";

cin >> records[sno].number;

sno++;

}

else

{

cout << "Phone directory is full." << endl;

}

}

void show(char sname[])

{

for (int i = 0; i<=sno-1; i++)

{

if (strcmp(records[i].name,sname) == 0)

{

cout<<"Name: "<<records[i].name<<"\nNumber: "<<records[i].number << endl;

}

}

}

};

void main()

{

tel\_dir entry;

char nm[20];

int ch;

clrscr();

do

{

cout<<"1. Add records"<<endl;

cout<<"2. Show"<<endl;

cout<<"3. Exit"<<endl;

cout<<"Enter your choice: ";

cin>>ch;

switch (ch) {

case 1:

entry.add();

break;

case 2:

cout << "Enter name to search: ";

cin>>nm;

entry.show(nm);

break;

case 3:

break;

default:

cout<<"ERROR 72: !! Something went wrong...!!\n";

break;

}

}while(ch!=3);

getch();

}

**Q9).Create a base class shape use the class two store double type value that could be used to compare the area. A drive to specific classes called triangle and rectangle. From the base shape and a member in get data to the base class to initialize base data member and another function display area. 10.**

**<source code>**

#include<iostream.h>

#include<conio.h>

class shape

{

protected:

double x,y;

public:

void get(int a,int b)

{

x=a;

y=b;

}

virtual void display\_area()=0;

};

class rec:public shape

{

public:

void display\_area()

{

cout<<"\nx = "<< x <<", y = "<< y <<" ";

cout<<"\nArea of Rectangle : ";

cout<<x\*y;

}

};

class tri:public shape

{

public:

void display\_area()

{

cout<<"\nx = "<< x <<", y = "<< y <<" ";

cout<<"\nArea of Triangle : ";

cout<<0.5\*x\*y;

}

};

void main(){

shape \*ptr;

rec r1;

tri t1;

clrscr();

r1.get(1,3);

t1.get(6,0);

ptr=&r1;

ptr->display\_area();

ptr=&t1;

ptr->display\_area();

getch();

}

**Q10).Write Program to implement Stack Operations like PUSH, POP, PEEP, UPDATE and DISPLAY using class and object**

**<source code>**

#include<iostream.h>

#include<conio.h>

class Stack

{

protected:

int s[5];

int top,data,n;

public:

Stack() {

top = -1;

n=5;

}

void push(int val) {

if(top == n-1)

cout<<"Stack overflow.....";

else {

top++;

s[top]=val;

}

}

int pop() {

if(top==-1) {

cout<<"stack underflow....";

return 0;

}

else {

data = s[top];

top--;

return data;

}

}

void display()

{ int i;

cout<<"\nprinting stack..."<<endl;

for(i=top;i>-1;i--)

cout<<s[i]<<"\t";

}

void peep(int position) {

int i;

i=position;

if(top-i+1<0)

cout<<"Stack underflow...";

else

cout<<"Data is: "<<s[top-i+1];

}

void edit(int position) {

int i;

i=position;

if(top-i+1<0)

cout<<"stack underflow..";

else {

cout<<"value at this location is: "<<s[top-i+1];

cout<<"\nEnter value to change: ";

cin>>s[top-i+1];

}

}

};

void main()

{

int ch,data;

Stack s;

clrscr();

do

{

cout<<"\n1. Push\n2. pop\n3. Display\n4. peep\n5. Upadte\n6. Exit";

cout<<"\nEnter your choice: ";

cin>>ch;

switch (ch)

{

case 1:

cout<<"Enter value to push: ";

cin>>data;

s.push(data);

break;

case 2:

cout<<s.pop()<<" is deleted...";

break;

case 3:

s.display();

break;

case 4:

cout<<"Enter from top to display: ";

cin>>data;

s.peep(data);

break;

case 5:

cout<<"Enter from top to update: ";

cin>>data;

s.edit(data);

break;

case 6:

break;

}

}while(ch!=6);

getch();

}

**Q11). Write Program to convert Infix to Postfix Expression using class and object.**

**<source code>**

#include<iostream.h>

#include<conio.h>

int top = -1;

int s[50];

class Stack

{

char data;

public:

void push(char val)

{

data = val;

top++;

s[top] = data;

}

char pop()

{

data = s[top];

top--;

return data;

}

int priority(char op)

{

int c=0;

switch (op)

{

case '^':

c=3;

break;

case '\*':case '/':

c=2;

break;

case '+':case '\_':

c=1;

break;

}

return c;

}

};

void main()

{

Stack si;

char in[50],post[50];

int j=0;

clrscr();

cout<<"Enter an Infix expression: ";

cin>>in;

for(int i=0;in[i]!='\0';i++)

{

switch(in[i])

{

case '(':

si.push(in[i]);

break;

case '+': case '-': case '\*': case '/':

while(si.priority(s[top])>=si.priority(in[i]))

post[j++]=si.pop();

si.push(in[i]);

break;

case ')':

while(s[top]!='(')

post[j++] = si.pop();

si.pop();

break;

default:

post[j++] = in[i];

break;

}

}

while(top>=0)

post[j++] = si.pop();

post[j] = '\0';

cout<<"\nPostfix: "<<post;

getch();

}

**Q12). Write Program to convert Infix to Prefix Expression using class and object.**

**<source code>**

#include<iostream.h>

#include<conio.h>

#include<string.h>

int top = -1;

int s[50];

class Stack

{

char data;

public:

void push(char val)

{

data = val;

top++;

s[top] = data;

}

char pop()

{

data = s[top];

top--;

return data;

}

int priority(char op)

{

int c=0;

switch (op)

{

case '^':

c=3;

break;

case '\*':case '/':

c=2;

break;

case '+':case '\_':

c=1;

break;

}

return c;

}

};

void main()

{

Stack si;

char in[50],pre[50];

int j=0;

clrscr();

cout<<"Enter an Infix expression: ";

cin>>in;

strrev(in);

for(int i=0;in[i]!='\0';i++)

{

switch(in[i])

{

case ')':

si.push(in[i]);

break;

case '+': case '-': case '\*': case '/':

while(si.priority(s[top])>=si.priority(in[i]))

pre[j++]=si.pop();

si.push(in[i]);

break;

case '(':

while(s[top]!=')')

pre[j++] = si.pop();

si.pop();

break;

default:

pre[j++] = in[i];

break; }

}

while(top>=0)

pre[j++] = si.pop();

pre[j] = '\0';

strrev(pre);

cout<<"\nPrefix: "<<pre;

getch();

}

**Q13). Write Program to implement Simple Queue Operations like Insert, Delete and Display.**

**<source code>**

#include<iostream.h>

#include<conio.h>

const int n = 5;

class Queue {

private:

int q[n];

int f, r;

public:

Queue() {

f = r = -1;

} void push(int data) {

if (r == n - 1) {

cout << "Overflow" << endl;

return;

}

r++;

q[r] = data;

if (f == -1)

f++;

}

int pop() {

int data;

if (f == -1) {

cout << "Underflow" << endl;

return 0;

}

data = q[f];

if (f == r)

f = r = -1;

else

f++;

return data;

}

void display() {

int i;

if (f == -1) {

cout << "Underflow" << endl;

return;

}

for (i = f; i <= r; i++) {

cout << q[i] << "\t";

}

cout << endl;

}

};

int main() {

Queue que;

int ch, data;

clrscr();

do {

cout << endl << "1. Push" << endl;

cout << "2. Pop" << endl;

cout << "3. Display" << endl;

cout << "4. Exit" << endl;

cout << "Enter your choice : ";

cin >> ch;

switch (ch) {

case 1:

cout << endl << "Enter data : ";

cin >> data;

que.push(data);

break;

case 2:

data = que.pop();

cout << endl << "Deleted data is : " << data;

break;

case 3:

que.display();

break;

}

} while (ch != 4);

getch();

return 0;

}

**Q14).Write Program to implement Circular Queue Operations like Insert, Delete and Display using class and object**

**<source code>**

#include<iostream.h>

#include<conio.h>

const int n = 5;

class CircularQueue

{

protected:

int q[n];

int f, r;

public:

CircularQueue()

{

f = r = -1;

}

void push(int data)

{

if ((r + 1) % n == f)

{

cout << "Overflow..." << endl;

return;

}

if (r == n - 1)

r = 0;

else

r++;

q[r] = data;

if (f == -1)

f++;

}

int pop()

{

int data;

if (f == -1)

{

cout << "Underflow..." << endl;

return 0;

}

data = q[f];

if (f == r)

f = r = -1;

else if (f == n - 1)

f = 0;

else

f++;

return data;

}

void display()

{

int i;

if (f == -1)

{

cout << "Underflow..." << endl;

return;

}

if (f <= r)

{

for (i = f; i <= r; i++)

{

cout << q[i] << "\t";

}

}

else

{

for (i = f; i < n; i++)

cout << q[i] << "\t";

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

cout << q[i] << "\t";

}

cout << endl;

}

};

void main() {

CircularQueue cq;

int ch, data;

clrscr();

do {

cout << endl << "1. Push" << endl;

cout << "2. Pop" <<endl;

cout << "3. Display" <<endl;

cout << "4. Exit" <<endl;

cout << "Enter your choice : ";

cin >> ch;

switch (ch)

{

case 1:

cout<<endl<<"Enter data : ";

cin>>data;

cq.push(data);

break;

case 2:

data=cq.pop();

cout<<endl<< "Deleted data is : " << data;

break;

case 3:

cq.display();

break;}

} while (ch != 4);

getch();

}

**Q15).Write Program to implement Double Ended Queue Operations like Insert, Delete and Display using class and object(To Perform Input Restricted)**

**<source code>**

#include <iostream.h>

#include<conio.h>

class Cir\_queue

{

protected:

int r, f;

int dq[5];

int n;

public:

Cir\_queue(int size)

{

n = size;

r = -1;

f = -1;

}

void insert\_r(int data)

{

if (r == n - 1)

{

cout << "Overflow..." << endl;

return;

}

r++;

dq[r] = data;

if (f == -1)

{

f = 0;

}

}

void insert\_l(int data)

{

if (f == 0)

{

cout << "Overflow..." << endl;

return;

}

if (f == -1)

f = r = n - 1;

else

f--;

dq[f] = data;

}

void display()

{

int i;

if (f == -1)

{

cout << "Underflow..." << endl;

return;

}

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

{

cout << dq[i] << "\t";

}

cout << endl;

}

};

void main()

{

int ch, data;

char io, ioside, side;

Cir\_queue rdq(5);

cout << "Input Restricted : " << endl;

cout << "Enter restricted side (L/R) : ";

cin >> ioside;

do

{

cout << "\n1. Insert " << endl;

cout << "2. Display " << endl;

cout << "3. Exit " << endl;

cout << "Enter your choice : ";

cin >> ch;

switch (ch) {

case 1:

cout << "Enter data : ";

cin >> data;

if (io == 'i') {

if (ioside == 'l') {

rdq.insert\_r(data);

}

else

rdq.insert\_l(data);

}

else {

cout << "Enter side (L/R) : ";

cin >> side;

if (side == 'r')

rdq.insert\_r(data);

else

rdq.insert\_l(data);

}

break;

case 2:

rdq.display();

break;

}

} while (ch != 3);

return 0;

}

Q16).Write Program to implement Double Ended Queue Operations like Insert, Delete and Display using class and object(To Perform Output Restricted)

<source code>

#include <iostream.h>

#include<conio.h>

class CircularQueue {

private:

int r, f;

int dq[5];

int n;

public:

CircularQueue(int size) {

n = size;

r = -1;

f = -1;

}

void insert\_r(int data) {

if (r == n - 1) {

cout << "Overflow..." << endl;

return;

}

r++;

dq[r] = data;

if (f == -1) {

f = 0;

}

}

int delete\_l() {

int data;

if (f == -1) {

cout << "Underflow..." << endl;

return 0;

}

data = dq[f];

dq[f] = 0;

if (f == r) {

f = r = -1;

} else {

f++;

}

return data;

}

int delete\_r() {

int data;

if (r == -1) {

cout << "Underflow..." << endl;

return 0;

}

data = dq[r];

dq[r] = 0;

if (f == r)

f = r = -1;

else

r--;

return data;

}

void insert\_l(int data) {

if (f == 0) {

cout << "Overflow..." << endl;

return;

}

if (f == -1)

f = r = n - 1;

else

f--;

dq[f] = data;

}

void display() {

int i;

if (f == -1) {

cout << "Underflow...." << endl;

return;

}

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

cout << dq[i] << "\t";

}

cout << endl;

}

};

void main() {

int ch, data;

char io, ioside, side;

CircularQueue cq(5);

cout << "Output Restricted..." << endl;

cout << "Enter restricted side (L/R) : ";

cin >> ioside;

do {

cout << endl << "1. Insert" << endl;

cout << "2. Delete" << endl;

cout << "3. Display" << endl;

cout << "4. Exit" << endl;

cout << "Enter your choice : ";

cin >> ch;

switch (ch) {

case 1:

cout << "Enter data : ";

cin >> data;

if (io == 'i') {

if (ioside == 'l') {

cq.insert\_r(data);

} else {

cq.insert\_l(data);

}

} else {

cout << "Enter side (L/R) : ";

cin >> side;

if (side == 'r') {

cq.insert\_r(data);

} else {

cq.insert\_l(data);

}

}

break;

case 2:

if (io == '0') {

if (ioside == 'l') {

data = cq.delete\_r();

} else {

data = cq.delete\_l();

}

} else {

cout << "Enter side (L/R) : ";

cin >> side;

if (side == 'r') {

data = cq.delete\_r();

} else {

data = cq.delete\_l();

}

}

break;

case 3:

cq.display();

break;

}

} while (ch != 4);

getch();

}