**PRACTICAL 1 =**

1) Write a Python program to store marks scored in subject “Fundamental of Data Structure” by N students

in the class. Write functions to compute following:

a) The average score of class

b) Highest score and lowest score of class

c) Count of students who were absent for the test

d) Display mark with highest frequency'''

marks = []

n = int(input("Enter the number of students in the class: "))

for i in range(n):

mark = int(input("Enter the mark scored by student {}: ".format(i+1)))

marks.append(mark)

def average\_score():

sum\_marks = 0

for mark in marks:

sum\_marks += mark

return sum\_marks/n

def highest\_score():

return max(marks)

def lowest\_score():

return min(marks)

def absent\_count():

return marks.count(-1)

def highest\_frequency():

frequency\_dict = {}

for mark in marks:

if mark in frequency\_dict:

frequency\_dict[mark] += 1

else:

frequency\_dict[mark] = 1

highest\_frequency = max(frequency\_dict.values())

for mark, frequency in frequency\_dict.items():

if frequency == highest\_frequency:

return mark

print("The average score of the class is:", average\_score())

print("The highest score of the class is:", highest\_score())

print("The lowest score of the class is:", lowest\_score())

print("Number of students who were absent for the test:", absent\_count())

print("The mark with the highest frequency is:", highest\_frequency())

**PRACTICAL 2-**

Q. Write a Python program that computes the net amount of a bank account based a transaction log **from console input. The transaction log format is shown as following**: D 100 W 200 (Withdrawal

net\_amount = 0

**while** True:

str = raw\_input ("Enter transaction: ")

transaction = str.split(" ")

type = transaction [0]

amount = int (transaction [1])

**if** type=="D" **or** type=="d":

net\_amount += amount

**elif** type=="W" **or** type=="w":

net\_amount -= amount

**else**:

**pass**

str = raw\_input ("want to continue (Y for yes) : ")

**if not** (str[0] =="Y" **or** str[0] =="y") :

**break**

**print** "Net amount: ", net\_amount

Output

Enter transaction: D 10000

want to **continue** (Y **for** yes) : Y

Enter transaction: W 5000

want to **continue** (Y **for** yes) : Y

Enter transaction: D 2000

want to **continue** (Y **for** yes) : Y

Enter transaction: W 100

want to **continue** (Y **for** yes) : N

Net amount: 6900

**PRACTICAL 3 =**

practical no 3-Write a Pythonprogram for department library which has N books, write

functions for following:

a)Delete the duplicate entries

b) Display books in ascending order based on cost of books

c) Count number of books with cost more than 500.

d)Copy books in a new list which has cost less than 500."""

# Department library class

class DepartmentLibrary:

def \_\_init\_\_(self, books):

self.books = books

# Delete duplicate entries

def delete\_duplicates(self):

self.books = list(set(self.books))

# Display books in ascending order based on cost

def display\_books\_ascending(self):

self.books.sort(key=lambda x: x[1])

for book in self.books:

print(book[0], book[1])

# Count number of books with cost more than 500

def count\_costly\_books(self):

count = 0

for book in self.books:

if book[1] > 500:

count += 1

return count

# Copy books with cost less than 500 to a new list

def copy\_affordable\_books(self):

affordable\_books = []

for book in self.books:

if book[1] < 500:

affordable\_books.append(book)

return affordable\_books

# Create an object of the DepartmentLibrary class

library = DepartmentLibrary([('Book1', 200), ('Book2', 300), ('Book3', 400), ('Book4', 500), ('Book5', 600)])

# Delete duplicate entries

library.delete\_duplicates()

# Display books in ascending order based on cost

library.display\_books\_ascending()

# Count number of books with cost more than 500

count = library.count\_costly\_books()

print(count)

# Copy books with cost less than 500 to a new list

affordable\_books = library.copy\_affordable\_books()

print(affordable\_books)

**PRACTICAL NO 4**

Q4.) Write a python program to store roll numbers of student array who

attended training program in sorted order.

Write function for searching whether particular student attended training

program or not, using Binary search and Fibonacci search'''

# Student class

class Student:

def \_\_init\_\_(self, roll\_num):

self.roll\_num = roll\_num

# Training program class

class TrainingProgram:

def \_\_init\_\_(self, students):

self.students = sorted(students)

# Binary search function

def binary\_search(self, roll\_num):

low = 0

high = len(self.students) - 1

while low <= high:

mid = (low + high) // 2

if self.students[mid].roll\_num == roll\_num:

return True

elif self.students[mid].roll\_num < roll\_num:

low = mid + 1

else:

high = mid - 1

return False

# Fibonacci search function

def fibonacci\_search(self, roll\_num):

fib2 = 0

fib1 = 1

fib = fib2 + fib1

while fib < len(self.students):

fib2 = fib1

fib1 = fib

fib = fib2 + fib1

offset = -1

while fib > 1:

i = min(offset+fib2, len(self.students)-1)

if self.students[i].roll\_num < roll\_num:

fib = fib1

fib1 = fib2

fib2 = fib - fib1

offset = i

elif self.students[i].roll\_num > roll\_num:

fib = fib2

fib1 = fib1 - fib2

fib2 = fib - fib1

else:

return True

if fib1 and self.students[offset+1].roll\_num == roll\_num:

return True

return False

# Create a list of Student objects

students = [Student(1), Student(2), Student(3), Student(4), Student(5)]

# Create an object of the TrainingProgram class

program = TrainingProgram(students)

# Search for a student using the binary search function

result = program.binary\_search(3)

print(result)

# Search for a student using the fibonacci search function

result = program.fibonacci\_search(3)

print(result)

**PRACTICAL NO.5**

Write a python program to store roll numbers of student array who

attended training program in sorted order.Write function for searching whether particular student attended training

program or not, using Binary search and Fibonacci search'''

nos=int(input("enter no of students"))

start=0

end=nos-1

present=[]

print("enter present student Roll NO ")

for i in range(0,nos):

b=int(input())

present.append(b)

key=int(input("enter roll no you want to search"))

flag=False

def binary(present1,start,end):

if(start<=end):

mid=int((start+end)/2)

if(key==present1[mid]):

print("Roll no found at position",mid)

flag=True

elif(key>present1[mid]):

binary(present,mid+1,end)

elif(key<present1[mid]):

binary(present1,start,mid-1)

else:

print("roll no not found")

binary(present,start,end)

**PRACTICAL NO 6**

Exp.6) Write a python program to store names and mobile numbers of your friends in

sorted order on names.

Search your friend from list using Fibonacci search. Insert friend if not present in

phonebook.'''

# Friend class

class Friend:

def \_\_init\_\_(self, name, mobile\_num):

self.name = name

self.mobile\_num = mobile\_num

# Phonebook class

class Phonebook:

def \_\_init\_\_(self, friends):

self.friends = sorted(friends, key=lambda x: x.name)

# Fibonacci search function

def fibonacci\_search(self, name):

fib2 = 0

fib1 = 1

fib = fib2 + fib1

while fib < len(self.friends):

fib2 = fib1

fib1 = fib

fib = fib2 + fib1

offset = -1

while fib > 1:

i = min(offset+fib2, len(self.friends)-1)

if self.friends[i].name < name:

fib = fib1

fib1 = fib2

fib2 = fib - fib1

offset = i

elif self.friends[i].name > name:

fib = fib2

fib1 = fib1 - fib2

fib2 = fib - fib1

else:

return self.friends[i]

if fib1 and self.friends[offset+1].name == name:

return self.friends[offset+1]

return None

# Insert friend function

def insert\_friend(self, friend):

self.friends.append(friend)

self.friends = sorted(self.friends, key=lambda x: x.name)

# Create a list of Friend objects

friends = [Friend('Alice', '123-456-7890'), Friend('Bob', '234-567-8901'), Friend('Eve', '345-678-9012')]

# Create an object of the Phonebook class

phonebook = Phonebook(friends)

# Search for a friend using the fibonacci search function

friend = phonebook.fibonacci\_search('Bob')

print(friend.mobile\_num)

# Insert a new friend

phonebook.insert\_friend(Friend('Charlie', '456-789-0123'))

# Search for the new friend

friend = phonebook.fibonacci\_search('Charlie')

print(friend.mobile\_num)

**PRACTICAL NO 7**

Department of Computer Engineering has student's club named 'Pinnacle Club'.

Students of second, third and final year of department can be granted membership on

request. Similarly one may cancel the membership of club. First node is reserved for

president of club and last node is reserved for secretary of club. Write C++ program to

maintain club member‘s information using singly linked list. Store student PRN and

Name. Write functions to:

a) Add and delete the members as well as president or even secretary.

b) Compute total number of members of club

c) Display members

d) Two linked lists exists for two divisions. Concatenate two lists.

CODE-

#include <iostream>

#include <string>

struct Member {

std::string name;

int prn;

Member\* next;

};

class Club {

private:

Member\* head; // Head of the linked list

Member\* tail; // Tail of the linked list

public:

Club();

~Club();

void addMember(std::string name, int prn);

void deleteMember(int prn);

int totalMembers();

void displayMembers();

void concatenate(Club& other);

};

Club::Club() : head(nullptr), tail(nullptr) {}

Club::~Club() {

// Delete all the nodes in the linked list

Member\* current = head;

while (current) {

Member\* next = current->next;

delete current;

current = next;

}

}

void Club::addMember(std::string name, int prn) {

Member\* newMember = new Member{name, prn, nullptr};

if (tail) {

// Add the new member to the end of the list

tail->next = newMember;

tail = newMember;

} else {

// The list is empty, so the new member is both the head and the tail

head = newMember;

tail = newMember;

}

}

void Club::deleteMember(int prn) {

Member\* current = head;

Member\* previous = nullptr;

while (current) {

if (current->prn == prn) {

// We found the member to delete

if (previous) {

// The member to delete is not the head, so update the next pointer of the previous member

previous->next = current->next;

} else {

// The member to delete is the head, so update the head pointer

head = current->next;

}

if (current == tail) {

// The member to delete is the tail, so update the tail pointer

tail = previous;

}

delete current;

return;

}

previous = current;

current = current->next;

}

std::cout << "Member with PRN " << prn << " not found\n";

}

int Club::totalMembers() {

int count = 0;

Member\* current = head;

while (current) {

count++;

current = current->next;

}

return count;

}

void Club::displayMembers() {

Member\* current = head;

while (current) {

std::cout << current->name << " (" << current->prn << ")\n";

current = current->next;

}

}

void Club::concatenate(Club& other) {

if (tail) {

// Add the other club's members to the end of this club's list

tail->next = other

**PRACTICAL 8**

/\*Pizza parlour accepting maximum M orders. Orders are served in first come first

served basis. Order once placed cannot be cancelled. Write C++ program to

simulate the system using circular queue using array. \*/

#include<iostream>

using namespace std;

class pizza{

int order[10];

int max;

int f,r;

public :

pizza(){

f=-1,r=-1;

cout<<"\nEnter Maximum order : ";

cin>>max;

}

int full(){

if(((f==0)&&(r==(max-1)))||(f==(r+1)%max))

return 1;

else

return 0;

}

int qempty(){

if(f==-1)

return 1;

else

return 0;

}

void add(int a){

if(full()){

cout<<"\nOrder is Full!!!";

}

else{

if(f==-1){

f=r=0;

}

else{

r=(r+1)%max;

}

order[r]=a;

}

}

void remove(){

int i;

i=order[f];

if(f==r){

f=r=-1;

}

else{

f=(f+1)%max;

}

cout<<"\n Order deleted : "<<i;

}

void display(){

int temp;

temp=f;

if(qempty())

{

cout<<"\nNo orders currently\n";

}

else{

cout<<"\nThe oders are : \n\n";

while(temp!=r){

cout<<" "<<order[temp];

temp=(temp+1)%max;

}

cout<<" "<<order[temp];

}}

};

int main(){

int ch;

pizza p;

do{

cout<<"\n1. Order \n2. Remove order \n3.Display orders \n4. Exit";

cin>>ch;

switch(ch){

case 1:int o;

cout<<"\nEnter Order number : ";

cin>>o;

p.add(o);

break;

case 2:p.remove();

break;

case 3:p.display();

break;

}

}while(ch!=4);

return 0;

}

**Practical 9**

/\*A palindrome is a string of character that‘s the same forward and backward.

Typically, punctuation, capitalization, and spaces are ignored. For example,

“Poor Dan is in a droop” is a palindrome, as can be seen by examining the

characters “poor danisina droop” and observing that they are the same forward

and backward. One way to check for a palindrome is to reverse the characters

in the string and then compare with them the original-in a palindrome, the

sequence will be identical. Write C++ program with functions

a) To print original string followed by reversed string using stack

b) To check whether given string is palindrome or not \*/

#include<iostream>

#include<string.h>

#define max 50

using namespace std;

class STACK

{

private:

char a[max];

int top;

public:

STACK()

{

top=-1;

}

void push(char);

void reverse();

void convert(char[]);

void palindrome();

};

void STACK::push(char c)

{

top++;

a[top] = c;

a[top+1]='\0';

}

void STACK::reverse()

{

char str[max];

cout<<"\n\nReverse string is : ";

for(int i=top,j=0; i>=0; i--,j++)

{

cout<<a[i];

str[j]=a[i];

}

cout<<endl;

}

void STACK::convert(char str[])

{

int j,k,len = strlen(str);

for(j=0, k=0; j<len; j++)

{

if( ( (int)str[j] >= 97 && (int)str[j] <=122 ) || ( (int)str[j] >= 65 && (int)str[j] <=90 ))

{

if( (int)str[j] <=90 )

{

str[k] = (char)( (int)str[j] + 32 );

}else

{

str[k] = str[j];

}

k++;

}

}

str[k]='\0';

cout<<endl<<"Converted String : "<<str<<"\n";

}

void STACK::palindrome()

{

char str[max];

int i,j;

for(i=top,j=0; i>=0; i--,j++)

{

str[j]=a[i];

}

str[j]='\0';

if(strcmp(str,a) == 0)

cout<<"\n\nString is palindrome";

else

cout<<"\n\nString is not palindrome";

}

int main()

{

STACK stack;

char str[max];

int i=0;

cout<<"\nEnter string to be reversed and check is it palindrome or not : \n\n";

cin.getline(str , 50);

stack.convert(str);

while(str[i] != '\0')

{

stack.push(str[i]);

i++;

}

stack.palindrome();

stack.reverse();

}

**Practical 10-**

Q.) The ticket booking system of Cinemax theater has to be implemented using C++ program. There are  10 rows and 7 seats in each row. Doubly circular linked list has to be maintained to keep track of free  seats at rows. Assume some random booking to start with. Use array to store pointers (Head pointer) to  each row. On demand

a) The list of available seats is to be displayed

b) The seats are to be booked

c) The booking can be cancelled.

#include

<iostream>

#include<stdlib.h>

using namespace std;

class node

{ public:

node\* next;

node\* prev;

int seat;

string id;

int status;

};

class cinemax

{

public:

node\* head,\* tail ,\* temp;

cinemax()

{

head=NULL;

}

void create\_list();

void display();

void book();

void cancel();

void avail();

};

void cinemax::create\_list()

{

int i=1;

temp=new node;

temp->seat=1;

temp->status=0;

temp->id="null";

tail=head=temp;

for(int i=2;i<=70;i++)

{

node \*p;

p= new node;

p->seat=i;

p->status=0;

p->id="null";

tail->next=p;

p->prev=tail;

tail=p;

tail->next=head;

head->prev=tail;

}

}

void cinemax::display()

{

{ int r=1;

node\* temp;

temp=head;

int count=0;

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

cout<<" Screen

this way \n";

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

while(temp->next!=head)

{

if(temp->seat/10==0)

cout<<"S0"<<temp->seat<<" :";

else

cout<<"S"<<temp->seat<<" :";

if(temp->status==0)

cout<<"|\_\_\_| ";

else

cout<<"|\_B\_| ";

count++;

if(count%7==0)

{

cout<<endl;

r++;

}

temp=temp->next;

}

cout<<"S"<<temp->seat<<" :";

if(temp->status==0)

cout<<"|\_\_\_| ";

else

cout<<"|\_B\_| ";

}

}

void cinemax::book()

{ int x;

string y;

label:

cout<<"\n\n\nEnter seat number to be booked\n"; cin>>x;

cout<<"Enter your ID number\n";

cin>>y;

if(x<1||x>70)

{

cout<<"Enter correct seat number to book (1-70)\n"; goto label;

}

node \*temp;

temp=new node;

temp=head;

while(temp->seat!=x)

{

temp=temp->next;

}

if(temp->status==1)

cout<<"Seat already booked!\n";

else{

temp->status=1;

temp->id=y;

cout<<"Seat "<<x<<" booked!\n";

}

}

void cinemax::cancel()

{

int x;

string y;

label1:

cout<<"Enter seat number to cancel booking\n";

cin>>x;

cout<<"Enter you ID\n";

cin>>y;

if(x<1||x>70)

{

cout<<"Enter correct seat number to cancel (1-70)\n";

goto label1;

}

node \*temp;

temp=new node;

temp=head;

while(temp->seat!=x)

{

temp=temp->next;

}

if(temp->status==0)

{

cout<<"Seat not booked yet!!\n";

}

else

{

if(temp->id==y)

{

temp->status=0;

cout<<"Seat Cancelled!\n";

}

else

cout<<"Wrong User ID !!! Seat cannot be cancelled!!!\n";

}

}

void cinemax::avail()

{

int r=1;

node\* temp;

temp=head;

int count=0;

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

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

cout<<" Screen this way \n";

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

while(temp->next!=head)

{

{

if(temp->seat/10==0)

cout<<"S0"<<temp->seat<<" :";

else

cout<<"S"<<temp->seat<<" :";

if(temp->status==0)

cout<<"|\_\_\_| ";

else if(temp->status==1)

cout<<" ";

count++;

if(count%7==0)

{

cout<<endl;

}

}

temp=temp->next;

}

if(temp->status==0)

{

cout<<"S"<<temp->seat<<" :";

if(temp->status==0)

cout<<"|\_\_\_| ";

}

}

int main()

{ cinemax obj;

obj.create\_list();

int ch;

char c='y';

while(c=='y')

{ obj.display();

cout<<"\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

cout<<" CINEMAX MOVIE THEATRE\n";

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

cout<<"\nEnter Choice\n1.Current SeatStatus\n2.Book Seat \n3.Available  Seat\n4.CancelSeat\n";

cin>>ch;

switch(ch)

{

case 1:obj.display();

break;

case 2: obj.book();

break;

case 3:obj.avail();

break;

case 4: obj.cancel();

break;

default: cout<<"Wrong choice input\n";

}

cout<<"\nDo you want to perform any other operation : (y/n)\n"; cin>>c;

}

return 0;

}

**PRACTICAL 11-**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

A double-ended queue (deque) is a linear list in which additions

and deletions may be made at either end. Obtain a data representation mapping  a deque into a onedimensional array. Write C++ program to simulate deque with functions  to add and delete elements from either end of the deque.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

using namespace std;

int queue[10];int front=-1;int rear=-1;

int size=10;

void display()

{

 cout<<"Status of Double Endded queue : ";

 for(int i=front;i<=rear;i++)

 {

 cout<<" "<<queue[i]<<"---->";

 }

}

int overflow()

{

 if(rear>=size)

 {

 return 1;

 }

 else

 {

 return 0;

 }

}

int isEmpty()

{

 if(front==-1 || front>rear)  {

 return 1;

 }

 else

 {

 return 0;

 }

}

void EnqueueF()

{

 int val1;

 cout<<"enter Value to insert in queue"<<endl;  cin>>val1;

 int p=isEmpty();

 if (p==1)

 {

 front=front+1;

 rear=rear+1;

 queue[front]=val1;

 display();

 }

 else

 {

 int m=overflow();

 if(m==1)

 {

 cout<<"Doble Ended queue is overflow"<<endl;  }

 else

 {

 for(int i=rear;i>=0;i--)

 {

 queue[i+1]=queue[i];

 }

 queue[front]=val1;

 rear=rear+1;

 display();

 }

 }

}

void EnqueueR()

{

 int val2;

 cout<<"enter Value to insert in queue"<<endl;  cin>>val2;

 int p=isEmpty();

 if (p==1)

 {

 front=front+1;

 rear=rear+1;

 queue[rear]=val2;

 display();

 }

 else

 {

 int m=overflow();

 if(m==1)

 {

 cout<<"Double Ended queue is overflow"<<endl;  }

 else

 {

 rear=rear+1;

 queue[rear]=val2;

 display();

 }

 }

}

void DequeueF()

{

 int s=isEmpty();

 if(s==1)

 {

 cout<<"\*\*\*\*\* Double Ended queue is empty \*\*\*\*\*\*"<<endl;  }

 else

 {

 cout<<"value"<<" "<<queue[front]<<" "<<"Deleted successfully"<<endl;  front=front+1;

 display();

 }

}

void DequeueR()

{

 int s=isEmpty();

 if(s==1)

 {

 cout<<"\*\*\*\*\* Double Ended queue is empty \*\*\*\*\*\*"<<endl;  }

 else

 {

 cout<<"value"<<" "<<queue[rear]<<" "<<"Deleted successfully"<<endl;  rear=rear-1;

 display();

 }

}

int main()

{

 int choice;

 char ch;

 do{

 cout<<"\*\*\*\*\*\*\*\*\* ENTER YOUR CHOICE \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;  cout<<" 1.Enqueue from front end"<<endl;

 cout<<" 2.Enqueue from rear end"<<endl;

 cout<<" 3.Dequeue from front end"<<endl;

 cout<<" 4.Dequeue from Rear end"<<endl;

 cin>>choice;

 switch(choice)

 {

 case 1:

 EnqueueF();

 break;

 case 2: EnqueueR();

 break;

 case 3: DequeueF();

 break;

 case 4: DequeueR();

 }

 cout<<endl<<"do you want to continue? Yes press 'y' for NO press 'n'";

 cin>>ch;

 }while(ch=='y');

 return 0;

}

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*OUTPUT OF PROGRAM \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\* ENTER YOUR CHOICE \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1.Enqueue from front end

2.Enqueue from rear end

3.Dequeue from front end

4.Dequeue from Rear end

1

enter Value to insert in queue

55

Status of Double Endded queue : 55---->

do you want to continue? Yes press 'y' for NO press 'n'y

\*\*\*\*\*\*\*\*\* ENTER YOUR CHOICE \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1.Enqueue from front end

2.Enqueue from rear end

3.Dequeue from front end

4.Dequeue from Rear end

1

enter Value to insert in queue

36

Status of Double Endded queue : 36----> 55----> do you want to continue? Yes press 'y' for NO press 'n'y \*\*\*\*\*\*\*\*\* ENTER YOUR CHOICE \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1.Enqueue from front end

2.Enqueue from rear end

3.Dequeue from front end

4.Dequeue from Rear end

2

enter Value to insert in queue

89

Status of Double Endded queue : 36----> 55----> 89----> do you want to continue? Yes press 'y' for NO press 'n'y \*\*\*\*\*\*\*\*\* ENTER YOUR CHOICE \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1.Enqueue from front end

2.Enqueue from rear end

3.Dequeue from front end

4.Dequeue from Rear end

2

enter Value to insert in queue

98

Status of Double Endded queue : 36----> 55----> 89----> 98----> do you want to continue? Yes press 'y' for NO press 'n'y \*\*\*\*\*\*\*\*\* ENTER YOUR CHOICE \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1.Enqueue from front end

2.Enqueue from rear end

3.Dequeue from front end

4.Dequeue from Rear end

3

value 36 Deleted successfully

Status of Double Endded queue : 55----> 89----> 98----> do you want to continue? Yes press 'y' for NO press 'n'y \*\*\*\*\*\*\*\*\* ENTER YOUR CHOICE \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1.Enqueue from front end

2.Enqueue from rear end

3.Dequeue from front end

4.Dequeue from Rear end

4

value 98 Deleted successfully

Status of Double Endded queue : 55----> 89----> do you want to continue? Yes press 'y' for NO press 'n'n

...Program finished with exit code 0

Press ENTER to exit console.\*/

**PRACTICAL 12-**

Q.) Queues are frequently used in computer programming, and a typical example is the creation of  a job queue by an operating system. If the operating system does not use priorities, then the  jobs are processed in the order they enter the system. Write C++ program for simulating job  queue. Write functions to add job and delete job from queue.

#include <iostream>

using namespace std;

int queue[10];int front=-1;int rear=-1;

int size=10;

void display()

{

 cout<<"status of job\_queue : ";

 for(int i=front;i<=rear;i++)

 {

 cout<<" "<<queue[i]<<"---->";

 }

}

int overflow()

{

 if(rear>=size)

 {

 return 1;

 }

 else

 {

 return 0;

 }

}

int isEmpty()

{

 if(front==-1 || front>rear)  {

 return 1;

 }

 else

 {

 return 0;

 }

}

void Enqueue()

{

 int job\_id;

 cout<<"enter job\_id"<<endl;  cin>>job\_id;

 int p=isEmpty();

 if (p==1)

 {

 front=front+1;

 rear=rear+1;

 queue[rear]=job\_id;  display();

 }

 else

 {

 int m=overflow();

 if(m==1)

 {

 cout<<"job\_queue is overflow"<<endl;

 }

 else

 {

 rear=rear+1;

 queue[rear]=job\_id;

 display();

 }

 }

}

void Dequeue()

{

 int s=isEmpty();

 if(s==1)

 {

 cout<<"\*\*\*\*\* job\_queue is empty \*\*\*\*\*\*"<<endl;

 }

 else

 {

 cout<<"job\_id"<<" "<<queue[front]<<" "<<"completed successfully"<<endl;  front=front+1;

 display();

 }

}

int main()

{

 int choice;

 char ch;

 do{

 cout<<"\*\*\*\*\*\*\*\*\* ENTER YOUR CHOICE \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;  cout<<" 1.Enqueue"<<endl;

 cout<<" 2.Dnqueue"<<endl;

 cin>>choice;

 switch(choice)

 {

 case 1:

 Enqueue();

 break;

 case 2: Dequeue();

 break;

 default: cout<<"Incorrect Choice ----- Try Again"<<endl;  }

 cout<<endl<<"do you want to continue? Yes press 'y' for NO press 'n'";  cin>>ch;

 }while(ch=='y');

 return 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* OUTPUT OF PROGRAM \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \* \*\*\*\*\*\*\*\*\* ENTER YOUR CHOICE \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1.Enqueue

2.Dnqueue

1

enter job\_id

5

status of job\_queue : 5---->

do you want to continue? Yes press 'y' for NO press 'n'y

\*\*\*\*\*\*\*\*\* ENTER YOUR CHOICE \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1.Enqueue

2.Dnqueue

1

enter job\_id

7

status of job\_queue : 5----> 7---->

do you want to continue? Yes press 'y' for NO press 'n'y

\*\*\*\*\*\*\*\*\* ENTER YOUR CHOICE \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1.Enqueue

2.Dnqueue

2

job\_id 5 completed successfully

status of job\_queue : 7---->

do you want to continue? Yes press 'y' for NO press 'n'y \*\*\*\*\*\*\*\*\* ENTER YOUR CHOICE \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1.Enqueue

2.Dnqueue

2

job\_id 7 completed successfully

status of job\_queue :

do you want to continue? Yes press 'y' for NO press 'n'y \*\*\*\*\*\*\*\*\* ENTER YOUR CHOICE \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1.Enqueue

2.Dnqueue

2

\*\*\*\*\* job\_queue is empty \*\*\*\*\*\*

do you want to continue? Yes press 'y' for NO press 'n'n

...Program finished with exit code 0

Press ENTER to exit console.\*/