1. # include<stdio.h>

# define MAX 5

int cqueue\_arr[MAX];

int front = -1;

int rear = -1;

void insert(int item)

{

if((front == 0 && rear == MAX-1) || (front == rear+1))

{

printf("Queue Overflow \n");

return;

}

if (front == -1) /\*If queue is empty \*/

{

front = 0;

rear = 0;

}

else

{

if(rear == MAX-1) /\*rear is at last position of queue \*/

rear = 0;

else

rear = rear+1;

}

cqueue\_arr[rear] = item ;

}

void del()

{

if (front == -1)

{

printf("Queue Underflow\n");

return ;

}

printf("Element deleted from queue is : %d\n",cqueue\_arr[front]);

if(front == rear) /\* queue has only one element \*/

{

front = -1;

rear=-1;

}

else

{

if(front == MAX-1)

front = 0;

else

front = front+1;

}

}

void display()

{

int front\_pos = front,rear\_pos = rear;

if(front == -1)

{

printf("Queue is empty\n");

return;

}

printf("Queue elements :\n");

if( front\_pos <= rear\_pos )

while(front\_pos <= rear\_pos)

{

printf("%d ",cqueue\_arr[front\_pos]);

front\_pos++;

}

else

{

while(front\_pos <= MAX-1)

{

printf("%d ",cqueue\_arr[front\_pos]);

front\_pos++;

}

front\_pos = 0;

while(front\_pos <= rear\_pos)

{

printf("%d ",cqueue\_arr[front\_pos]);

front\_pos++;

}

}

printf("\n");

}

int main()

{

int choice,item;

do

{

printf("1.Insert\n");

printf("2.Delete\n");

printf("3.Display\n");

printf("4.Quit\n");

printf("Enter your choice : ");

scanf("%d",&choice);

switch(choice)

{

case 1 :

printf("Input the element for insertion in queue : ");

scanf("%d", &item);

insert(item);

break;

case 2 :

del();

break;

case 3:

display();

break;

case 4:

break;

default:

printf("Wrong choice\n");

}

}while(choice!=4);

return 0;

}

**2. C++ implementation of the approach**

#include <bits/stdc++.h>

using namespace std;

void countPermutations(int N, int B)

{

int x = pow(B, N);

int y = pow(B, N - 1);

cout << x - y << "\n";

}

int main()

{

int N = 6;

int B = 4;

countPermutations(N, B);

return 0;

}

3. #include <iostream>

#include <stdlib.h>

using namespace std;

class Stack {

private:

static const int max = 100;

int arr[max];

int top;

public:

Stack() { top = -1; }

bool isEmpty();

bool isFull();

int pop();

void push(int x);

};

bool Stack::isEmpty()

{

if (top == -1)

return true;

return false;

}

bool Stack::isFull()

{

if (top == max - 1)

return true;

return false;

}

int Stack::pop()

{

if (isEmpty()) {

cout << "Stack Underflow";

abort();

}

int x = arr[top];

top--;

return x;

}

void Stack::push(int x)

{

if (isFull()) {

cout << "Stack Overflow";

abort();

}

top++;

arr[top] = x;

}

class SpecialStack : public Stack {

Stack min;

public:

int pop();

void push(int x);

int getMin();

};

void SpecialStack::push(int x)

{

if (isEmpty() == true) {

Stack::push(x);

min.push(x);

}

else {

Stack::push(x);

int y = min.pop();

min.push(y);

if (x < y)

min.push(x);

else

min.push(y);

}

}

int SpecialStack::pop()

{

int x = Stack::pop();

min.pop();

return x;

}

int SpecialStack::getMin()

{

int x = min.pop();

min.push(x);

return x;

}

int main()

{

SpecialStack s;

s.push(10);

s.push(20);

s.push(30);

cout << s.getMin() << endl;

s.push(5);

cout << s.getMin();

return 0;

}