**Function:**

**Int element,topA=-1,stack[max],topB=max**

**PushA(){**

if(topA==topB){

Printf(“stack is an overflow”);

}

else{

Printf(“enter the element”);

Scanf(“%d”,&element);

topA++;

Stack[topA]=element;

}

**}**

**popA(){**

if(topA==-1){

printf(“stack is an Underflow”);

}

else{

element=stack[topA];

topA--;

}

**}**

**traverseA(){**

if(topA==-1){

printf(“stack is an Underflow”);

}

else{

printf(“element of stack.”);

for(i=topA;i>=0;i--){

printf(“%d”);

}

}

**}**

**pushB(){**

if(topB==topA){

Printf(“stack is an overflow”);

}

else{

Printf(“enter the element”);

Scanf(“%d”,&element);

topB--;

Stack[topA]=element;

}

**}**

**popB(){**

if(topB==max){

printf(“stack is an Underflow”);

}

else{

element=stack[topB];

topB++;

}

**}**

**TraverseB(){**

if(topB==max){

printf(“stack is an Underflow”);

}

else{

printf(“Element of stackB.”);

for(i=topB;i<max;i++){

printf(“%d”);

}

}

**}**

**Assignment**

Write a menu driven program to implement multiple stack operation.

**Queue**

Insert element(enqueue)🡪Rear pointer

Delete element(dequeue)🡪Front pointer

Traverse()

**Memory Representation of Queue**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A[0] | A[1] | A[2] | A[3] | A[4] |

**Rear=-1,Front=-1**

**underflow condition:**

**if(front==Rear)**

**Overflow Condition:**

If(Rear==max-1)

**Insert 50**

**Rear++=0**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 50(Rear) |  |  |  |  |

Rear++=1

Insert 70

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 50 | 70(Rear) |  |  |  |

Rear ++=2

Insert 60

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 50 | 70 | 60(Rear) |  |  |

Front++=0

Delete 50

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ~~50(Front)~~ | 70 | 60(Rear) |  |  |

Front++=1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | ~~70(Front)~~ | ~~60(Rear)(front)~~ |  |  |

Int I,element,rear=-1,front=-1,queue[max];

**Enqueue(){**

If(rear==max-1){

Printf(“Queue if is overflow”);

}

Else{

Printf(“enter the element of Queue”);

Scanf(“%d”,&element);

Rear++;

Queue[Rear]=element;

}

}