

# DATA STRUCTURES &ALGORITHMS

## ASSIGNMENT-2

**NAME:**MD.Amrath

**ID:**B181094

**CLASS:**AB2- 305

1. Write a program that implement stack (its operations) using  
i) Arrays      ii) Linked list(Pointers).

Operations :

push(), pop(), display(), stackOverflow(), stackUnderflow()

### **i)Arrays**

**Psuedo code:**

Begin

create a array with size

take variable = -1

#### INSERTION OF DATA (PUSH)

input data

IF top>n-1 then stackoverflow

Else if top=-1 then insert the data at index 0

Else increment top and insert data into array stack[top]

#### DELETION OF DATA

IF top ==-1 then stackunderflow

Else If top >n-1 then top ==-1

Else PRINT stack[top]

#### DISPLAY ELEMENTS

FOR i=top i<=0 i++

PRINT stack[i]

call fuctions in main

push()

pop()

display()

End

## Code:-

```
#include<stdio.h>

#include<stdlib.h>

#define MAX 5

int stack[MAX];

int top=-1;

void push()
{
    int element;
    printf("\nEnter element to insert: ");
    scanf("%d",&element);
    if(top>=MAX-1)
    {
        printf("stack overflow\n");
    }
    else
    {
        stack[++top]=element;
    }
}

void pop()
{
    if(top<0)
    {
        printf("stack underflow\n");
    }
}
```

```

        else
        {
            printf("\n%d is popped\n",stack[top--]);
        }
    }
}

void display()
{
    int i;
    printf("\nstatus of elements in stack: ");
    for(i=top;i>=0;i--)
    {
        printf("%d ",stack[i]);
    }
    printf("\n");
}

int main()
{
    int ch;
    while(1)
    {
        printf("\nstack operatios:-\n");
        printf("-----\n");
        printf(" 1.PUSH\n");
        printf(" 2.POP\n");
        printf(" 3.DISPLAY\n");
        printf(" 4.EXIT\n");
        printf("\nChoose any option: ");
        scanf("%d",&ch);
    }
}

```

```
switch(ch)
{
    case 1:
        push();
        break;
    case 2:
        pop();
        break;
    case 3:
        display();
        break;
    case 4:
        exit(0);
        break;
    default:
        printf("\ninvalid input\n");
}
}
return(0);
}
```

C:\Users\PETTAM RAKESH\Desktop\Data structures\Assignment\_2\stack\stack\_wth\_arrays.exe

```
stack operatios:-
-----
1.PUSH
2.POP
3.DISPLAY
4.EXIT
```

Choose any option: 1

enter element to insert: 22

```
stack operatios:-
-----
1.PUSH
2.POP
3.DISPLAY
4.EXIT
```

Choose any option: 1

enter element to insert: 33

```
stack operatios:-
-----
1.PUSH
2.POP
3.DISPLAY
4.EXIT
```

Choose any option: 1

enter element to insert: 44

```
stack operatios:-
-----
1.PUSH
2.POP
3.DISPLAY
4.EXIT
```

Choose any option: 1

enter element to insert: 55

```
stack operatios:-
-----
1.PUSH
2.POP
3.DISPLAY
4.EXIT
```

Choose any option: 1

enter element to insert: 66

```
stack operatios:-
-----
```



Type here to search



C:\Users\PETTAM RAKESH\Desktop\Data structures\Assignment\_2\stack\stack\_wth\_arrays.exe

```
stack operatios:-
-----
1.PUSH
2.POP
3.DISPLAY
4.EXIT

Choose any option: 1
stack overflow

stack operatios:-
-----
1.PUSH
2.POP
3.DISPLAY
4.EXIT

Choose any option: 3

status of elements in stack: 66 55 44 33 22

stack operatios:-
-----
1.PUSH
2.POP
3.DISPLAY
4.EXIT

Choose any option: 2

66 is popped

stack operatios:-
-----
1.PUSH
2.POP
3.DISPLAY
4.EXIT

Choose any option: 2

55 is popped

stack operatios:-
-----
1.PUSH
2.POP
3.DISPLAY
4.EXIT

Choose any option: 2

44 is popped

stack operatios:-
```



Type here to search



```
C:\Users\PETTAM RAKESH\Desktop\Data structures\Assignment_2\stack\stack_wth_arrays.exe
44 is popped
stack operatios:-
-----
1.PUSH
2.POP
3.DISPLAY
4.EXIT
Choose any option: 2
33 is popped
stack operatios:-
-----
1.PUSH
2.POP
3.DISPLAY
4.EXIT
Choose any option: 2
22 is popped
stack operatios:-
-----
1.PUSH
2.POP
3.DISPLAY
4.EXIT
Choose any option: 2
stack underflow
stack operatios:-
-----
1.PUSH
2.POP
3.DISPLAY
4.EXIT
Choose any option: 3
status of elements in stack:
Stack is empty
stack operatios:-
-----
1.PUSH
2.POP
3.DISPLAY
4.EXIT
Choose any option: 4
-----
Process exited after 22.77 seconds with return value 0
Press any key to continue . . .
```

## ii)Linked List

## Pseudo code:

Begin

define a struct node with data and next (struct node type)

### PUSH--INSEERTION

push(int x)

Create a newnode using malloc

Store newnode->data=x

newnode->next=top

top=newnode

### POP--DELETION

Create temp Variable

temp=top

IF top==0

PRINT .....UNDERFLOW condition

Else

PRINT DELETED DATA =temp->data

Change top=temp->next

free(temp)

PEEK ELEMENT

PRINT Front>data

### DISPLAY

Create variable struct node\*temp

IF top==0

PRINT stack is empty

ELSE

Assign temp=top;

While temp!=0

PRINT temp->data

Increment temp=temp->next

Call functions: push(x) pop() display()



End

## CODE:

```
#include<stdio.h>
#include<stdlib.h>
struct node{
    int data;
    struct node *next;
};
struct node *top=NULL;
struct node *create_node(int element){
    struct node *nn;
    nn=(struct node*)malloc(sizeof(struct node));
    nn->data=element;
    nn->next=NULL;
    return nn;
}
void push(){
    int element;
    struct node *nn;
    printf("\nEnter element to insert: ");
    scanf("%d",&element);
    nn=create_node(element);
    if(top==NULL)
    {
        top=nn;
    }
    else
    {
        nn->next=top;
        top=nn;
    }
}
void pop(){
    struct node *temp=top;
    if(top==NULL)
    {
        printf("\nStack underflow\n");
    }
    else if(top->next==NULL)
    {
        top=NULL;
        free(temp);
    }
}
```

```

        else
        {
            top=top->next;
            free(temp);
        }
    }
void display()
{
    struct node *temp=top;
    printf("\nstatus of stack elements: ");
    if(top== NULL)
    {
        printf("Stack is empty\n");
    }
    else
    {
        while(temp!=NULL)
        {
            printf("%d ",temp->data);
            temp=temp->next;
        }
    }
    printf("\n");
}
int main()
{
    int ch;
    while(1)
    {
        printf("\nstack operatios:-\n");
        printf("-----\n");
        printf(" 1.PUSH\n");
        printf(" 2.POP\n");
        printf(" 3.DISPLAY\n");
        printf(" 4.EXIT\n");
        printf("\nChoose any option: ");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:
                push();
                break;
            case 2:
                pop();
                break;

```

```
        case 3:
            display();
            break;
        case 4:
            exit(0);
            break;
        default:
            printf("\ninvalid input\n");
    }
}
return(0);
}
```

C:\Users\PETTAM RAKESH\Desktop\Data structures\Assignment\_2\stack\stack\_wth\_Linked\_list.exe

Stack operatios:-

-----  
1.PUSH  
2.POP  
3.DISPLAY  
4.EXIT

Choose any option: 1

enter element to insert: 22

Stack operatios:-

-----  
1.PUSH  
2.POP  
3.DISPLAY  
4.EXIT

Choose any option: 1

enter element to insert: 33

Stack operatios:-

-----  
1.PUSH  
2.POP  
3.DISPLAY  
4.EXIT

Choose any option: 1

enter element to insert: 44

Stack operatios:-

-----  
1.PUSH  
2.POP  
3.DISPLAY  
4.EXIT

Choose any option: 3

status of stack elements: 44 33 22

Stack operatios:-

-----  
1.PUSH  
2.POP  
3.DISPLAY  
4.EXIT

Choose any option: 2



Type here to search



```
C:\Users\PETTAM RAKESH\Desktop\Data structures\Assignment_2\stack\stack_wth_Linked_list.exe
Stack operatios:-
-----
1.PUSH
2.POP
3.DISPLAY
4.EXIT
Choose any option: 2
Stack operatios:-
-----
1.PUSH
2.POP
3.DISPLAY
4.EXIT
Choose any option: 2
Stack operatios:-
-----
1.PUSH
2.POP
3.DISPLAY
4.EXIT
Choose any option: 2
stack underflow
Stack operatios:-
-----
1.PUSH
2.POP
3.DISPLAY
4.EXIT
Choose any option: 3
status of stack elements: Stack is empty
Stack operatios:-
-----
1.PUSH
2.POP
3.DISPLAY
4.EXIT
Choose any option: 4
-----
Process exited after 15.12 seconds with return value 0
Press any key to continue . . .
```

2. **Write a program that implement Queue (its operations) using**

i) **Arrays**    ii) **Linked list(Pointers).**

Operations :

Enqueue(), Dequeue(), display(), QueueOverflow(),

QueueUnderflow()

## i)Arrays

### Pseudo code:

Begin

Define array queue[N]

Take 2 variables front=-1 rear=-1

### ENQUEUE:

function defination with arg x

IF front==-1 && rear==-1

Assign front=rear=0

store queue[rear]=a

Else If rear>=N-1

PRINT QueueOverflow

ELSE

Increment rear++

store queue[rear]=a

### DEQUEUE:

IF front== -1 || front>rear

PRINT QueueUnderflow

front=-1

Else

PRINT Deleted element: queue[front]

Increment front++

### DISPLAY

Take i

IF front == -1

PRINT Queue is empty

ELSE

FOR i=front;i<=rear;i++

PRINT queue[i]

Call required function

Enqueue(a)

dequeue()

display()

End

## CODE:

```
#include<stdio.h>
```

```
#include<stdlib.h>
```

```
#define MAX 5
```

```
int queue[MAX];
```

```
int front=-1;
```

```
int rear=-1;
```

```
void enq()
```

```
{
```

```
    int e;
```

```
    if(rear>=MAX-1)
```

```
    {
```

```
        printf("\nQueue overflow\n");
```

```
        return;
```

```
    }
```

```
    else
```

```
    {
```

```
        printf("\nEnter element to insert: ");
```

```
        scanf("%d",&e);
```

```
        if(front== -1 && rear== -1)
```

```
        {
```

```

        front=rear=0;
    }
    else
    {
        rear=rear+1;
    }
    queue[rear]=e;
}

}

void deq()
{
    if(front== -1 || front>rear)
    {
        printf("\nQueue underflow\n");
    }
    else
    {
        queue[front++];
    }
}

void display()
{
    int i;
    printf("\nStatus of elements in queue: ");
    if(front== -1 || front>rear )
    {
        printf("Queue is empty\n");
    }
}

```



```

    }
    else
    {
        for(i=front;i<=rear;i++)
        {
            printf("%d ",queue[i]);
        }
    }
    printf("\n");
}

int main()
{
    int ch;
    while(1)
    {
        printf("\nQueue operatios:-\n");
        printf("-----\n");
        printf(" 1.ENQUEUE\n");
        printf(" 2.DEQUEUE\n");
        printf(" 3.DISPLAY\n");
        printf(" 4.EXIT\n");
        printf("\nChoose any option: ");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:
                enq();
                break;

```

```
        case 2:
            deq();
            break;
        case 3:
            display();
            break;
        case 4:
            exit(0);
            break;
        default:
            printf("\ninvalid input\n");
    }
}
return(0);
}
```

C:\Users\PETTAM RAKESH\Desktop\Data structures\Assignment\_2\Queue\Queue\_wth\_arrars.exe

Queue operatios:-

- 
- 1.ENQUEUE
- 2.DEQUEUE
- 3.DISPLAY
- 4.EXIT

Choose any option: 1

enter element to insert: 22

Queue operatios:-

- 
- 1.ENQUEUE
- 2.DEQUEUE
- 3.DISPLAY
- 4.EXIT

Choose any option: 1

enter element to insert: 33

Queue operatios:-

- 
- 1.ENQUEUE
- 2.DEQUEUE
- 3.DISPLAY
- 4.EXIT

Choose any option: 1

enter element to insert: 44

Queue operatios:-

- 
- 1.ENQUEUE
- 2.DEQUEUE
- 3.DISPLAY
- 4.EXIT

Choose any option: 1

enter element to insert: 55

Queue operatios:-

- 



🔍 Type here to search



C:\Users\PETTAM RAKESH\Desktop\Data structures\Assignment\_2\Queue\Queue\_wth\_arrays.exe

Choose any option: 1

enter element to insert: 66

Queue operatios:-

- 1.ENQUEUE
- 2.DEQUEUE
- 3.DISPLAY
- 4.EXIT

Choose any option: 1

Queue overflow

Queue operatios:-

- 1.ENQUEUE
- 2.DEQUEUE
- 3.DISPLAY
- 4.EXIT

Choose any option: 3

Status of elements in queue: 22 33 44 55 66

Queue operatios:-

- 1.ENQUEUE
- 2.DEQUEUE
- 3.DISPLAY
- 4.EXIT

Choose any option: 2

Queue operatios:-

- 1.ENQUEUE
- 2.DEQUEUE
- 3.DISPLAY
- 4.EXIT

Choose any option: 3

Status of elements in queue: 33 44 55 66

Queue operatios:-



Type here to search



C:\Users\PETTAM RAKESH\Desktop\Data structures\Assignment\_2\Queue\Queue\_wth\_arrays.exe

Queue operatios:-

- 1.ENQUEUE
- 2.DEQUEUE
- 3.DISPLAY
- 4.EXIT

Choose any option: 2

Queue operatios:-

- 1.ENQUEUE
- 2.DEQUEUE
- 3.DISPLAY
- 4.EXIT

Choose any option: 3

Status of elements in queue: 44 55 66

Queue operatios:-

- 1.ENQUEUE
- 2.DEQUEUE
- 3.DISPLAY
- 4.EXIT

Choose any option: 2

Queue operatios:-

- 1.ENQUEUE
- 2.DEQUEUE
- 3.DISPLAY
- 4.EXIT

Choose any option: 2

Queue operatios:-

- 1.ENQUEUE
- 2.DEQUEUE
- 3.DISPLAY
- 4.EXIT

Choose any option: 2



Type here to search



C:\Users\PETTAM RAKESH\Desktop\Data structures\Assignment\_2\Queue\Queue\_wth\_arrars.exe

Queue operatios:-

- 1.ENQUEUE
- 2.DEQUEUE
- 3.DISPLAY
- 4.EXIT

Choose any option: 2

Queue underflow

Queue operatios:-

- 1.ENQUEUE
- 2.DEQUEUE
- 3.DISPLAY
- 4.EXIT

Choose any option: 3

Status of elements in queue: Queue is empty

Queue operatios:-

- 1.ENQUEUE
- 2.DEQUEUE
- 3.DISPLAY
- 4.EXIT

Choose any option: 4

Process exited after 61.39 seconds with return value 0

Press any key to continue . . .

## ii)Linked list(Pointers)

### Pseudo code:

Begin

Create struct node of int data struct node \*next;

Take struct node \*front=0 \*rear=0

#### ENQUEUE:

Function enq int a

Create struct node \*newnode

memory allocation newnode=(struct node\*)malloc(sizeof(struct node

assign newnode->data=a and newnode->next=0;

IF front==0 && rear==0

front=rear=newnode

ELSE

rear->next=newnode

rear=newnode

#### DEQUEUE:

CREATE struct node\*temp

temp=front

IF front==0

PRINT QueueUnderflow

Else

PRINT DELETED DATA AS front->data

front=front->next;

IF front==0

rear=0

free(temp)

PEEK

IIF front==0 && rear==0

PRINT QueueUnderflow

Else

PRINT front->data

#### DISPLAY

Create struct node\*temp;

IF front==0 && rear==0

PRINT QUEUE IS EMPTY

CALL fuctions

Enqueue(a) dequeue() display()

End

## CODE:

```
#include<stdio.h>
```

```
#include<stdlib.h>
```

```
struct node{
```

```
    int data;
```

```
    struct node *next;
```

```
};
```

```
struct node *front=NULL;
```

```
struct node *rear=NULL;
```

```
struct node *createnode(int element)
```

```
{
```

```
    struct node *nn;
```

```
    nn=(struct node *)malloc(sizeof(struct node));
```

```
    nn->data=element;
```

```
    nn->next=NULL;
```

```
    return nn;
```

```
}
```

```
void enq()
```

```
{
```

```
    int e;
```

```
    printf("\nenter element to insert: ");
```



```
scanf("%d",&e);
struct node *nn;
nn=createnode(e);
if(front==NULL && rear==NULL)
{
    front=rear=nn;
}
else
{
    rear->next=nn;
    rear=nn;
}
}
```

```
void deq()
{
    struct node *temp=front;
    if(front==NULL)
    {
        printf("Queue underflow\n");
    }
    else
    {
        front=front->next;
        if(front==NULL)
        {
            rear=NULL;
        }
    }
}
```

```

        free(temp);
    }
}

void display()
{
    struct node *temp=front;
    printf("\nStatus of elements in queue: ");
    if(temp==NULL)
    {
        printf("Queue is empty\n");
        return;
    }
    while(temp!=NULL)
    {
        printf("%d ",temp->data);
        temp=temp->next;
    }
    printf("\n");
}

int main()
{
    int ch;
    while(1)
    {
        printf("\nQueue operatios:-\n");
        printf("-----\n");
        printf(" 1.ENQUEUE\n");
    }
}

```

```
printf(" 2.DEQUEUE\n");
printf(" 3.DISPLAY\n");
printf(" 4.EXIT\n");
printf("\nChoose any option: ");
scanf("%d",&ch);
switch(ch)
{
    case 1:
        enq();
        break;
    case 2:
        deq();
        break;
    case 3:
        display();
        break;
    case 4:
        exit(0);
        break;
    default:
        printf("\ninvalid input\n");
}
}
return(0);
}
```

C:\Users\PETTAM RAKESH\Desktop\Data structures\Assignment\_2\Queue\queue\_wth\_linked\_list.exe

Queue operatios:-

- 1.ENQUEUE
- 2.DEQUEUE
- 3.DISPLAY
- 4.EXIT

Choose any option: 1

enter element to insert: 22

Queue operatios:-

- 1.ENQUEUE
- 2.DEQUEUE
- 3.DISPLAY
- 4.EXIT

Choose any option: 1

enter element to insert: 33

Queue operatios:-

- 1.ENQUEUE
- 2.DEQUEUE
- 3.DISPLAY
- 4.EXIT

Choose any option: 1

enter element to insert: 44

Queue operatios:-

- 1.ENQUEUE
- 2.DEQUEUE
- 3.DISPLAY
- 4.EXIT

Choose any option: 3

Status of elements in queue: 22 33 44



Type here to search



C:\Users\PETTAM RAKESH\Desktop\Data structures\Assignment\_2\Queue\queue\_wth\_linked\_list.exe

Queue operatios:-

- 1.ENQUEUE
- 2.DEQUEUE
- 3.DISPLAY
- 4.EXIT

Choose any option: 2

Queue operatios:-

- 1.ENQUEUE
- 2.DEQUEUE
- 3.DISPLAY
- 4.EXIT

Choose any option: 3

Status of elements in queue: 33 44

Queue operatios:-

- 1.ENQUEUE
- 2.DEQUEUE
- 3.DISPLAY
- 4.EXIT

Choose any option: 2

Queue operatios:-

- 1.ENQUEUE
- 2.DEQUEUE
- 3.DISPLAY
- 4.EXIT

Choose any option: 2

Queue operatios:-

- 1.ENQUEUE
- 2.DEQUEUE
- 3.DISPLAY
- 4.EXIT



Type here to search



C:\Users\PETTAM RAKESH\Desktop\Data structures\Assignment\_2\Queue\queue\_wth\_linked\_list.exe

Queue operatios:-

- 1.ENQUEUE
- 2.DEQUEUE
- 3.DISPLAY
- 4.EXIT

Choose any option: 2

Queue underflow

Queue operatios:-

- 1.ENQUEUE
- 2.DEQUEUE
- 3.DISPLAY
- 4.EXIT

Choose any option: 3

Status of elements in queue: Queue is empty

Queue operatios:-

- 1.ENQUEUE
- 2.DEQUEUE
- 3.DISPLAY
- 4.EXIT

Choose any option: 4

Process exited after 30.15 seconds with return value 0

Press any key to continue . . .

### 3. Write a program that implement Circular Queue using arrays

Operations :

Enqueue(), Dequeue(), display(), QueueOverflow(), QueueUnderflow()

#### Psuedo code:

Begin

Define a array queue[N]

Take front=rear=-1

### ENQUEUE

func ENQ int X

IF (rear+1)%N==front)

PRINT QueueOverflow

IF ELSE front== -1 && rear== -1

front=rear=0

queue[rear]=a

ELSE

rear=(rear+1)%N

queue[rear]=a

### DEQUEUE

IF front== -1 && rear== -1

PRINT QueueUnderflow

Else IF front==rear

PRINT Deleted element:queue[front]

front=rear=-1

ELSE

Deleted element:queue[front]

front=(front+1)%N

### DISPLAY

TAKE and Assign i=front

IF front == -1

PRINT QueueUnderflow

ELSE

do

PRINT queue[i])

i=(i+1)%N

While i!=(rear+1)%N

call fuctions

End

## **CODE:**

```
#include<stdio.h>
#include<stdlib.h>
#define max 5
int cqueue[max];
int front=-1;
int rear=-1;
void enq()
{
    int e;
    if(((rear+1)%max)==front)
    {
        printf("Queue overflow\n");
        return;
    }
    else
    {
        printf("\nenter element to insert: ");
        scanf("%d",&e);
        if(front==-1)
        {
            front=rear=0;
        }
        else
        {
            rear=(rear+1)%max;
        }
        cqueue[rear]=e;
    }
}
```



```

}
void deq()
{
    if(front==-1)
    {
        printf("Queue underflow\n");
        return;
    }
    else
    {
        if(front==rear)
        {
            front=rear=-1;
        }
        else
        {
            front=(front+1)%max;
        }
    }
}

void display()
{
    int i;
    printf("\nStatus of elements in queue: ");
    if(front==-1)
    {
        printf("queue is empty\n");
        return;
    }
    for(i=front;i!=rear;i=(i+1)%max)
    {

```

```

        printf("%d ",cqueue[i]);
    }
    printf("%d\n",cqueue[i]);
}
int main()
{
    int ch;
    while(1)
    {
        printf("\nCircular Queue operatios:-\n");
        printf("-----\n");
        printf(" 1.ENQUEUE\n");
        printf(" 2.DEQUEUE\n");
        printf(" 3.DISPLAY\n");
        printf(" 4.EXIT\n");
        printf("\nChoose any option: ");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:
                enq();
                break;
            case 2:
                deq();
                break;
            case 3:
                display();
                break;
            case 4:
                exit(0);
                break;

```

default:

```
printf("\ninvalid input\n");
```

```
}
```

```
}
```

```
return(0);
```

```
}
```

C:\Users\PETTAM RAKESH\Desktop\Data structures\Assignment\_2\Queue\circular\_queue\_wth\_arrays.exe

Circular Queue operatios:-

- 1.ENQUEUE
- 2.DEQUEUE
- 3.DISPLAY
- 4.EXIT

Choose any option: 1

enter element to insert: 22

Circular Queue operatios:-

- 1.ENQUEUE
- 2.DEQUEUE
- 3.DISPLAY
- 4.EXIT

Choose any option: 1

enter element to insert: 33

Circular Queue operatios:-

- 1.ENQUEUE
- 2.DEQUEUE
- 3.DISPLAY
- 4.EXIT

Choose any option: 1

enter element to insert: 44

Circular Queue operatios:-

- 1.ENQUEUE
- 2.DEQUEUE
- 3.DISPLAY
- 4.EXIT

Choose any option: 1

enter element to insert: 55



Type here to search



C:\Users\PETTAM RAKESH\Desktop\Data structures\Assignment\_2\Queue\circular\_queue\_wth\_arrays.exe

Circular Queue operatios:-

- 1.ENQUEUE
- 2.DEQUEUE
- 3.DISPLAY
- 4.EXIT

Choose any option: 1

enter element to insert: 66

Circular Queue operatios:-

- 1.ENQUEUE
- 2.DEQUEUE
- 3.DISPLAY
- 4.EXIT

Choose any option: 1

Queue overflow

Circular Queue operatios:-

- 1.ENQUEUE
- 2.DEQUEUE
- 3.DISPLAY
- 4.EXIT

Choose any option: 3

Status of elements in queue: 22 33 44 55 66

Circular Queue operatios:-

- 1.ENQUEUE
- 2.DEQUEUE
- 3.DISPLAY
- 4.EXIT

Choose any option: 2

Circular Queue operatios:-



C:\Users\PETTAM RAKESH\Desktop\Data structures\Assignment\_2\Queue\circular\_queue\_wth\_arrays.exe

Circular Queue operatios:-

- 1.ENQUEUE
- 2.DEQUEUE
- 3.DISPLAY
- 4.EXIT

Choose any option: 1

enter element to insert: 10

Circular Queue operatios:-

- 1.ENQUEUE
- 2.DEQUEUE
- 3.DISPLAY
- 4.EXIT

Choose any option: 2

Circular Queue operatios:-

- 1.ENQUEUE
- 2.DEQUEUE
- 3.DISPLAY
- 4.EXIT

Choose any option: 3

Status of elements in queue: 44 55 66 10

Circular Queue operatios:-

- 1.ENQUEUE
- 2.DEQUEUE
- 3.DISPLAY
- 4.EXIT

Choose any option: 2

Circular Queue operatios:-

- 1.ENQUEUE
- 2.DEQUEUE



Type here to search



```
C:\Users\PETTAM RAKESH\Desktop\Data structures\Assignment_2\Queue\circular_queue_wth_array

Choose any option: 2
Circular Queue operatios:-
-----
1.ENQUEUE
2.DEQUEUE
3.DISPLAY
4.EXIT
Choose any option: 2
Circular Queue operatios:-
-----
1.ENQUEUE
2.DEQUEUE
3.DISPLAY
4.EXIT
Choose any option: 2
Circular Queue operatios:-
-----
1.ENQUEUE
2.DEQUEUE
3.DISPLAY
4.EXIT
Choose any option: 2
Circular Queue operatios:-
-----
1.ENQUEUE
2.DEQUEUE
3.DISPLAY
4.EXIT
Choose any option: 2
Queue underflow
Circular Queue operatios:-
-----
1.ENQUEUE
2.DEQUEUE
3.DISPLAY
4.EXIT
Choose any option: 3
Status of elements in queue: queue is empty
Circular Queue operatios:-
-----
1.ENQUEUE
2.DEQUEUE
3.DISPLAY
4.EXIT
Choose any option: 4
-----
Process exited after 272.2 seconds with return value 0
Press any key to continue . . .
```

4. Write a program that implement Priority Queue (its operations) using

- i) Arrays
- ii) Linked list(Pointers).

Psuedo code:

## Begin

### PEnqueue()

IF((Front == 0)&&(Rear == N-1))

PRINT "Overflow Condition"

Else

IF Front == -1

Front = Rear = 0

Queue[Rear] = Data

Priority[Rear] = Priority

ELSE IF(Rear == N-1)

FOR i=Front;i<=Rear;i++)

FOR(i=Front;i<=Rear;i++)

Q[i-Front] = Q[i]

Pr[i-Front] = Pr[i]

Rear = Rear-Front

Front = 0

FOR(i = r;i>f;i-)

IF p>Pr[i]

Q[i+1] = Q[i] Pr[i+1] = Pr[i]

ELSE

Q[i+1] = data Pr[i+1] = p

Rear++

### PDequeue()

IF Front == -1

PRINT "Queue Under flow condition"

ELSE

PRINT"Q[f],Pr[f]"



IF(Front==Rear)

Front = Rear = -1

ELSE

FRONT++

**Display**

FOR(i=Front;i<=Rear;i++)

PRINT(Q[i],Pr[i])

## CODE:

```
#include<stdio.h>
```

```
#include<stdlib.h>
```

```
#define N 3
```

```
int Q[N],Pr[N];
```

```
int r = -1,f = -1;
```

```
void Penq()
```

```
{
```

```
    int data,p;
```

```
    int i;
```

```
    if((f==0)&&(r==N-1)) //Check if Queue is full
```

```
        printf("Queue overflow");
```

```
    else
```

```
    {
```

```
        printf("\nenter the data: ");
```

```
        scanf("%d",&data);
```

```
        printf("\nenter priority of data: ");
```

```
        scanf("%d",&p);
```

```
        if(f==-1)//if Queue is empty
```

```
        {
```

```
            f = r = 0;
```

```
            Q[r] = data;
```

```
            Pr[r] = p;
```

```

}
else if(r == N-1)
{
    for(i=f;i<=r;i++) { Q[i-f] = Q[i]; Pr[i-f] = Pr[i]; r = r-f; f = 0; for(i = r;i>f;i--)
        {
            if(p>Pr[i])
            {
                Q[i+1] = Q[i];
                Pr[i+1] = Pr[i];
            }
            else
                break;
            Q[i+1] = data;
            Pr[i+1] = p;
            r++;
        }
    }
}
else
{
    for(i = r;i>=f;i--)
    {
        if(p>Pr[i])
        {
            Q[i+1] = Q[i];
            Pr[i+1] = Pr[i];
        }
        else
            break;
    }
    Q[i+1] = data;
    Pr[i+1] = p;
    r++;
}

```

```

        }
    }

}

void display() //print the data of Queue
{
    int i;
    if(f== -1 )
    {
        printf("\n PQueue is empty\n");
    }
    else{

        for(i=f;i<=r;i++)
        {
            printf("\nElement = %d\tPriority = %d",Q[i],Pr[i]);

        }
    }
}

int Pdeq()
{
    if(f == -1)
    {
        printf("Queue underflow");
    }
    else
    {
        printf("deleted Element = %d\t Its Priority = %d",Q[f],Pr[f]);
        if(f==r)
            f = r = -1;
        else
            f++;
    }
}

```

```

int main()
{
    int ch;
    while(1)
    {
        printf("\nQueue operatios:-\n");
        printf("-----\n");
        printf(" 1.PENQUEUE\n");
        printf(" 2.PDEQUEUE\n");
        printf(" 3.DISPLAY\n");
        printf(" 4.EXIT\n");
        printf("\nChoose any option: ");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:
                Penq();
                break;
            case 2:
                Pdeq();
                break;
            case 3:
                display();
                break;
            case 4:
                exit(0);
            default:
                printf("\ninvalid input\n");
        }
    }
    return(0);
}

```

C:\Users\PETTAM RAKESH\Desktop\p.exe

Queue operatios:-

- 1.PENQUEUE
- 2.PDEQUEUE
- 3.DISPLAY
- 4.EXIT

Choose any option: 1

enter the data: 44

enter priority of data: 6

Queue operatios:-

- 1.PENQUEUE
- 2.PDEQUEUE
- 3.DISPLAY
- 4.EXIT

Choose any option: 1

enter the data: 55

enter priority of data: 2

Queue operatios:-

- 1.PENQUEUE
- 2.PDEQUEUE
- 3.DISPLAY
- 4.EXIT

Choose any option: 1

enter the data: 9

enter priority of data: 9

Queue operatios:-

- 1.PENQUEUE
- 2.PDEQUEUE
- 3.DISPLAY
- 4.EXIT

Choose any option: 3

Element = 9      Priority = 9

Element = 44     Priority = 6

Element = 55     Priority = 2

Queue operatios:-



Type here to search



C:\Users\PETTAM RAKESH\Desktop\p.exe

```
-----  
1.PENQUEUE  
2.PDEQUEUE  
3.DISPLAY  
4.EXIT
```

Choose any option: 2

deleted Element = 9      Its Priority = 9

Queue operatios:-

```
-----  
1.PENQUEUE  
2.PDEQUEUE  
3.DISPLAY  
4.EXIT
```

Choose any option: 3

Element = 44      Priority = 6

Element = 55      Priority = 2

Queue operatios:-

```
-----  
1.PENQUEUE  
2.PDEQUEUE  
3.DISPLAY  
4.EXIT
```

Choose any option: 1

enter the data: 55

enter priority of data: 4

Queue operatios:-

```
-----  
1.PENQUEUE  
2.PDEQUEUE  
3.DISPLAY  
4.EXIT
```

Choose any option: 2

deleted Element = 44      Its Priority = 6

Queue operatios:-

```
-----  
1.PENQUEUE  
2.PDEQUEUE  
3.DISPLAY  
4.EXIT
```

Choose any option: 2

deleted Element = 44      Its Priority = 6

Queue operatios:-

```
-----  
1.PENQUEUE  
2.PDEQUEUE
```



Type here to search



```
C:\Users\PETTAM RAKESH\Desktop\p.exe
2.PDEQUEUE
3.DISPLAY
4.EXIT

Choose any option: 2
Queue underflow
Queue operatios:-
-----
1.PENQUEUE
2.PDEQUEUE
3.DISPLAY
4.EXIT

Choose any option: 3

PQueue is empty
Queue operatios:-
-----
1.PENQUEUE
2.PDEQUEUE
3.DISPLAY
4.EXIT

Choose any option: 4

-----
Process exited after 156.2 seconds with return value 0
Press any key to continue . . .
```

## ii)Linked list(Pointers)

## Psuedo code:

CREATE node

Take int data,priority

Take struct node \*next

Take , Assign struct node \*front=NULL

struct node \*temp

CREATE NODE:

struct node \*createnode int element int priority

Create struct node \*nn;

nn=(struct node\*)malloc(sizeof(struct node))

nn->data=element

nn->priority=priority

nn->next=NULL

### PEnqueue:

Function PEnqueue int e,int p

create struct node \*nn,\*temp=front

nn=createnode(e,p)

IF front==NULL || p < front->priority){nn->next=front

front=nn

Else

While temp->next!=NULL && temp->next->priority<= p

temp=temp->next

nn->next=temp->next

temp->next=nn

### PDEQUEUE:

IF front!=NULL

temp=front



PRINT Deleted data : front->data Priority data : front->priority

front=front->next

free(temp)

DISPLAY:

Create struct node \*temp=front;

PRINT Elements in Priority Queue

IF temp!=NULL

While temp!=NULL

PRINT temp->data,temp->priority

temp=temp->next

DISPLAY

Create and Assign struct node \*temp=front;

IF temp!=NULL

While temp!=NULL

PRINT temp->data,temp->priority

temp=temp->next

UNDERFLOW:

IF front==NULL

PRINT queue underflow

Call functions PEnqueue(13,1) PEnqueue(16,0) PEnqueue(98,4) display()

PDequeue() display() underflow()

End

## CODE:

```
#include<stdio.h>
```

```
#include<stdlib.h>
```

```
struct node{
```

```

    int data;
    int priority;
    struct node *next;
};

struct node *front=NULL;

struct node *createnode(int e,int p)
{
    struct node *nn;
    nn=(struct node *)malloc(sizeof(struct node));
    nn->data=e;
    nn->priority=p;
    nn->next=NULL;
    return nn;
}

void penq()
{
    struct node *nn,*temp=front;
    int element,priority;
    printf("\nEnter element to insert: ");
    scanf("%d",&element);
    printf("\nEnter priority of element: ");
    scanf("%d",&priority);
    nn=createnode(element,priority);
    if(front==NULL || priority<front->priority )
    {
        nn->next=front;
        front=nn;
    }
}

```

```

else
{
    while(temp->next!=NULL && temp->next->priority<=priority)
    {
        temp=temp->next;
    }
    nn->next=temp->next;
    temp->next=nn;
}
}

void pdeq()
{
    struct node *temp=front;
    if(front==NULL)
    {
        printf("\nQueue underflow");
    }
    else
    {
        front=front->next;
        free(temp);
    }
}

void display()
{
    struct node *temp=front;
    printf("\nStatus of elements in Queue: ");
    if(front==NULL)

```

```

    {
        printf("\nQueue is empty\n");
    }
    else
    {
        while(temp!=NULL)
        {
            printf("\n%d\t %d ",temp->data,temp->priority);
            temp=temp->next;
        }
        printf("\n");
    }
}

int main()
{
    int ch;
    while(1)
    {
        printf("\nQueue operatios:-\n");
        printf("-----\n");
        printf(" 1.PENQUEUE\n");
        printf(" 2.PDEQUEUE\n");
        printf(" 3.DISPLAY\n");
        printf(" 4.EXIT\n");
        printf("\nChoose any option: ");
        scanf("%d",&ch);
        switch(ch)
        {

```

```
        case 1:
            penq();
            break;
        case 2:
            pdeq();
            break;
        case 3:
            display();
            break;
        case 4:
            exit(0);
        default:
            printf("\ninvalid input\n");
    }
}
return(0);
}
```

C:\Users\PETTAM RAKESH\Desktop\Data structures\Assignment\_2\Queue\priority\_queue\_linedlist.exe

Queue operatios:-

- 1.PENQUEUE
- 2.PDEQUEUE
- 3.DISPLAY
- 4.EXIT

Choose any option: 1

enter element to insert: 22

enter priority of element: 1

Queue operatios:-

- 1.PENQUEUE
- 2.PDEQUEUE
- 3.DISPLAY
- 4.EXIT

Choose any option: 33

invalid input

Queue operatios:-

- 1.PENQUEUE
- 2.PDEQUEUE
- 3.DISPLAY
- 4.EXIT

Choose any option: 1

enter element to insert: 33

enter priority of element: 1

Queue operatios:-

- 1.PENQUEUE
- 2.PDEQUEUE
- 3.DISPLAY
- 4.EXIT

Choose any option: 1

enter element to insert: 66

enter priority of element: 5

Queue operatios:-

- 1.PENQUEUE
- 2.PDEQUEUE
- 3.DISPLAY
- 4.EXIT



Type here to search



C:\Users\PETTAM RAKESH\Desktop\Data structures\Assignment\_2\Queue\priority\_queue\_linkedlist.exe

Choose any option: 3

Status of elements in Queue:

22	1
33	1
66	5

Queue operatios:-

-----  
1.PENQUEUE  
2.PDEQUEUE  
3.DISPLAY  
4.EXIT

Choose any option: 2

Queue operatios:-

-----  
1.PENQUEUE  
2.PDEQUEUE  
3.DISPLAY  
4.EXIT

Choose any option: 3

Status of elements in Queue:

33	1
66	5

Queue operatios:-

-----  
1.PENQUEUE  
2.PDEQUEUE  
3.DISPLAY  
4.EXIT

Choose any option: 1

enter element to insert: 55

enter priority of element: 9

Queue operatios:-

-----  
1.PENQUEUE  
2.PDEQUEUE  
3.DISPLAY  
4.EXIT

Choose any option: 3

Status of elements in Queue:

33	1
66	5
55	9

Queue operatios:-



Type here to search



C:\Users\PETTAM RAKESH\Desktop\Data structures\Assignment\_2\Queue\priority\_queue\_linedlist.exe

Choose any option: 2

Queue operatios:-

-----  
1.PENQUEUE  
2.PDEQUEUE  
3.DISPLAY  
4.EXIT

Choose any option: 2

Queue operatios:-

-----  
1.PENQUEUE  
2.PDEQUEUE  
3.DISPLAY  
4.EXIT

Choose any option: 2

Queue operatios:-

-----  
1.PENQUEUE  
2.PDEQUEUE  
3.DISPLAY  
4.EXIT

Choose any option: 2

Queue underflow

Queue operatios:-

-----  
1.PENQUEUE  
2.PDEQUEUE  
3.DISPLAY  
4.EXIT

Choose any option: 3

Status of elements in Queue:

Queue is empty

Queue operatios:-

-----  
1.PENQUEUE  
2.PDEQUEUE  
3.DISPLAY  
4.EXIT

Choose any option: 4

-----  
Process exited after 55.9 seconds with return value 0

Press any key to continue . . .



Type here to search

