

Lab-4

```

1) Standard queues.
#include <stdio.h>
#include <stdlib.h>
#define size 5
int queue[size], front = -1, rear = -1;
void push(int a) {
    if (rear == size - 1) {
        printf("overflow condition\n");
        return;
    }
    else if (front == -1 && rear == -1) {
        front = 0;
        rear = 0;
    }
    else {
        rear = rear + 1;
    }
    queue[rear] = a;
}

int pop() { int n;
    if (front == -1 && rear == -1) {
        printf("underflow condition\n");
        return;
    }
    else if (front == 0 && rear == 0) {
        n = queue[front];
        front = 1; // reset queue
        rear = -1;
    }
    else {
        n = queue[front];
        front = front + 1;
    }
    return n;
}

```

```
void display() {
```

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

```
        printf("Queue is empty\n");  
        return;
```

```
    }
```

```
    else {
```

```
        int i;
```

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

```
        }
```

```
    }
```

```
}
```

```
void main() {
```

```
    int op, n;
```

```
    while printf("Enter 1. PUSH\n  
2. POP\n 3. Display\n 4. -1 to  
stop execution\n");
```

```
    while (1) {
```

```
        printf("Enter the operator\n");
```

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

```
        if (op == -1) {
```

```
            printf("Execution is  
stopped\n");
```

```
            break;
```

```
        }
```

```
        else {
```

```
            switch(op) {
```

```
                case 1: printf("Enter the  
number\n");
```

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

```
                push(n);
```

```
                break;
```



```

case 2: n = pop();
        printf("%d is popped\n", n);
        break;
case 3: printf("queue is:\n");
        display();
        break;
default: printf("Invalid choice\n");

```

```

    }
}
/
/
/

```

op

Enter the operator

1. enqueue

2. dequeue

3. display

4. -1 to stop

Enter the operator

1

Enter the no.

2

Enter operator

2

2 is dequeued

```

2) circular queue
#include <stdio.h>
#include <stdlib.h>
#define size 5
int queue[size], front = -1, rear = -1;
void push ( int a ) {
    if ( ( front == 0 && rear == size - 1 ) ||
        front == rear + 1 ) {
        printf ( " Overflow condition \n " );
        return ;
    }
    else if ( front == -1 && rear == -1 ) {
        front = 0;
        rear = 0;
    }
    else {
        rear = ( rear + 1 ) % size;
    }
    q[rear] = a;
}

int pop () { int a;
    if ( front == -1 ) {
        printf ( " Underflow condition \n " );
        return ;
    }
    else if ( front == rear ) {
        a = queue[front];
        front = -1;
        rear = -1;
    }
    else {
        a = queue[front];
        front = ( front + 1 ) % size;
    }
}

```



```
return a;
```

```
}
```

```
void display()
```

```
{
    if (front == -1) {
```

```
        printf("Queue is empty\n");
```

```
        return;
```

```
    }
```

```
    else {
        int i;
```

```
        for (i = front; i != rear; i++
```

```
            i = (i+1) % size);
```

```
            printf("%d\t", queue[i]);
```

```
        }
```

```
        printf("%d\t", queue[i]);
```

```
    }
```

```
}
```

```
void main() {
```

```
    int op, n;
```

```
    printf("Enter 1. PUSH In 2. POP In
```

```
    3. Display In 4. -1 to stop execution\n");
```

```
    while (1) {
```

```
        printf("Enter the operator\n");
```

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

```
        if (op == -1) {
```

```
            printf("Execution is stopped\n");
```

```
            break;
```

```
        }
```

```
        else {
            switch (op) {
```

```
                case 1: printf("Enter the
```

```
                number\n");
```

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

```
push(n);
```

```
break;
```

```
case 2: n = pop();
```

```
printf("%d is popped", n);
```

```
break;
```

```
case 3: printf("queue is 1");
```

```
display();
```

```
break;
```

```
default: printf("Invalid choice");
```

```
break;
```

```
}
```

```
}
```

```
}
```

```
}
```

gph
ok

Enter 1 Enqueue

2. Dequeue

3. Display

4. -1 to stop execution

Enter operators

2

Queue is empty underflow

-1 is dequed

Enter operators

1

Enter no

2

Both
o/p same

Standard queues output:

C:\Users\bmscel\Desktop\18M22CS291\standardq.exe

```
Enter the operation:
1.enqueue
2.dequeue
3.display
4.-1 to stop
Enter the operator
2
queue underflow
Enter the operator
1
Enter the number:
2
successfully enqueued
Enter the operator
1
Enter the number:
3
successfully enqueued
Enter the operator
1
Enter the number:
4
successfully enqueued
Enter the operator
1
Enter the number:
5
successfully enqueued
Enter the operator
1
Enter the number:
6
successfully enqueued
Enter the operator
1
Enter the number:
7
queue overflow
Enter the operator
3
Elements are:
2
3
4
5
6
Enter the operator
2
deleted element is=2
Enter the operator
```

Circular queues output:

"C:\C LAB PROGRAMS 243\circularq.exe"

```
Enter 1.Enqueue
2.Dequeue
3.Display
4.-1 to stop execution

Enter operator
2
Queue is empty/underflow
-1 is Dequeued

Enter operator
1
Enter no
2

Enter operator
1
Enter no
3

Enter operator
1
Enter no
4
Queue is full/overflow

Enter operator
3
Queue : 2      3
Enter operator
2
2 is Dequeued

Enter operator
-1

Process returned -1 (0xFFFFFFFF)   execution time : 26.744 s
Press any key to continue.
```