Date 08/01/2024
Page

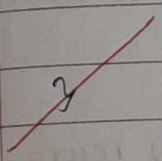
Lab-4

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-1;
    }
    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;
    printf("Enter 1. PUSH\n 2. POP\n 3. Display\n 4. -1 to
     stop execution\n");
    while(1){
        printf("\nEnter 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 poped\n"
                ,n);
        break;
case 3: display();
        break;
default: printf("Invalid
                choise\n");
}
}
}
```



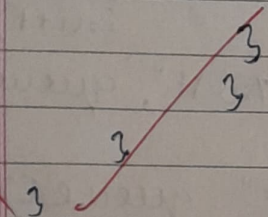
```
op
Enter the operator
1. enqueue
2. dequeue
3. displays
4. -1 to stop
Enter the operator
1
Enter the no.
2
Enter operator
2
2 is dequeued
```

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=-1;
        }
        else{
                rear = (rear+1) %.size;
        }
        q[rear] = n;
}

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

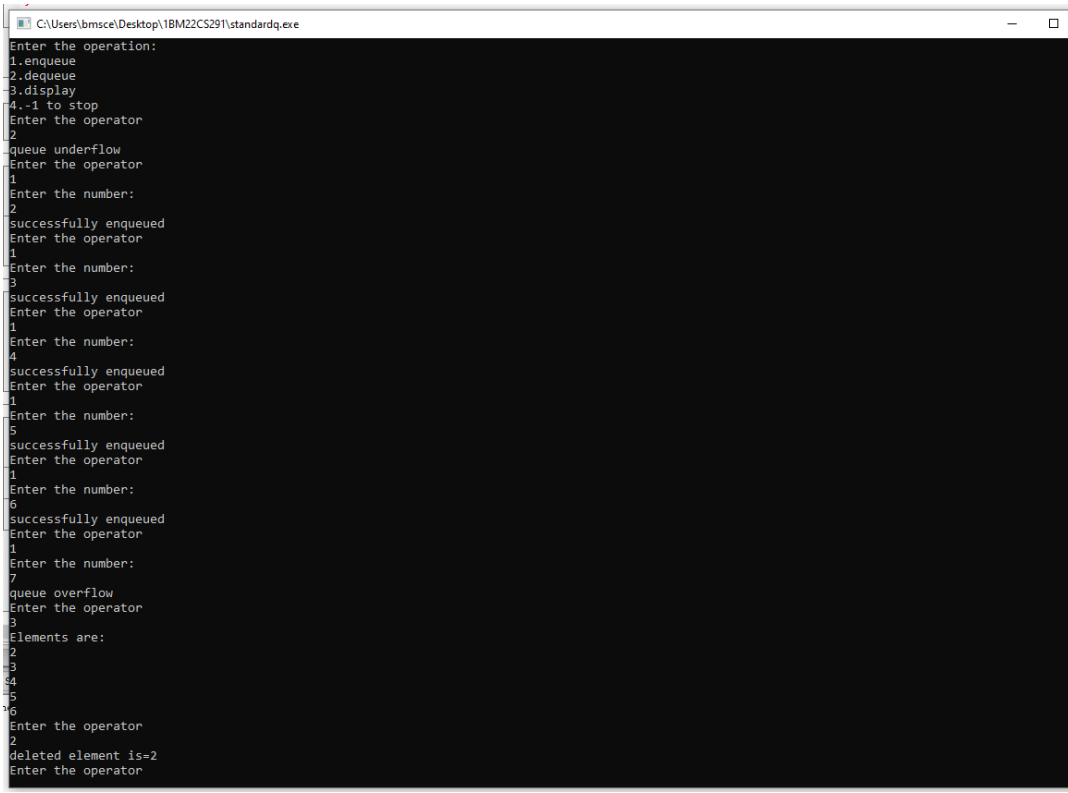
```
    return a;
}

void display(
    if (front == -1){
        printf("Queue is empty \n");
        return;
    }
    else { int i;
        for(i=front; i!=rear; i=(i+1)%.size){
            printf("%.d \t", queue[i]);
        }
        printf("%.d \t", queue[i]);
    }
}

void main(){
    int op, n;
    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);
        break;
case 3: printf("queue is \t");
        display();
        break;
default: printf("invalid choice\n");
        break;
}
}
}
```



```
Enter 1Enqueue
2. Dequeue
3. Display
4. -1 to stop execution
Enter operator
2
Queue is empty (underflow
-1 is decued
Enter operator
1
Enter no
2
```

Standard queues output:



Circular queues output: