Sanjana Shetty

3-E

a) WAP to simulate the working of a queue of integers using an array. Provide the following operations: Insert, Delete, Display. The program should print appropriate messages for queue empty and queue overflow conditions

b) WAP to simulate the working of a circular queue of integers using an array. Provide the following operations: Insert, Delete & Display

The program should print appropriate messages for queue empty and queue overflow conditions

a)

```
#include <stdio.h>
#include <stdlib.h>
#define N 4

int q[N];
int REAR = -1;
int FRONT = -1;

void enq();
void deq();
void display();
```

```
void enq() {
   if (REAR == N - 1) {
      printf("Overflow!\n");
   } else {
      int item;
      printf("Enter the element to insert:\n");
      scanf("%d", &item);
      if (REAR == -1 && FRONT == -1) {
         REAR++;
         q[REAR]=item;
         FRONT++;
      }
      else{
      REAR++;
      q[REAR] = item;
      }
   }
}

void deq() {
   int val;
   if (FRONT == -1 || FRONT > REAR) {
      printf("Queue empty!\n");
   } else {
```

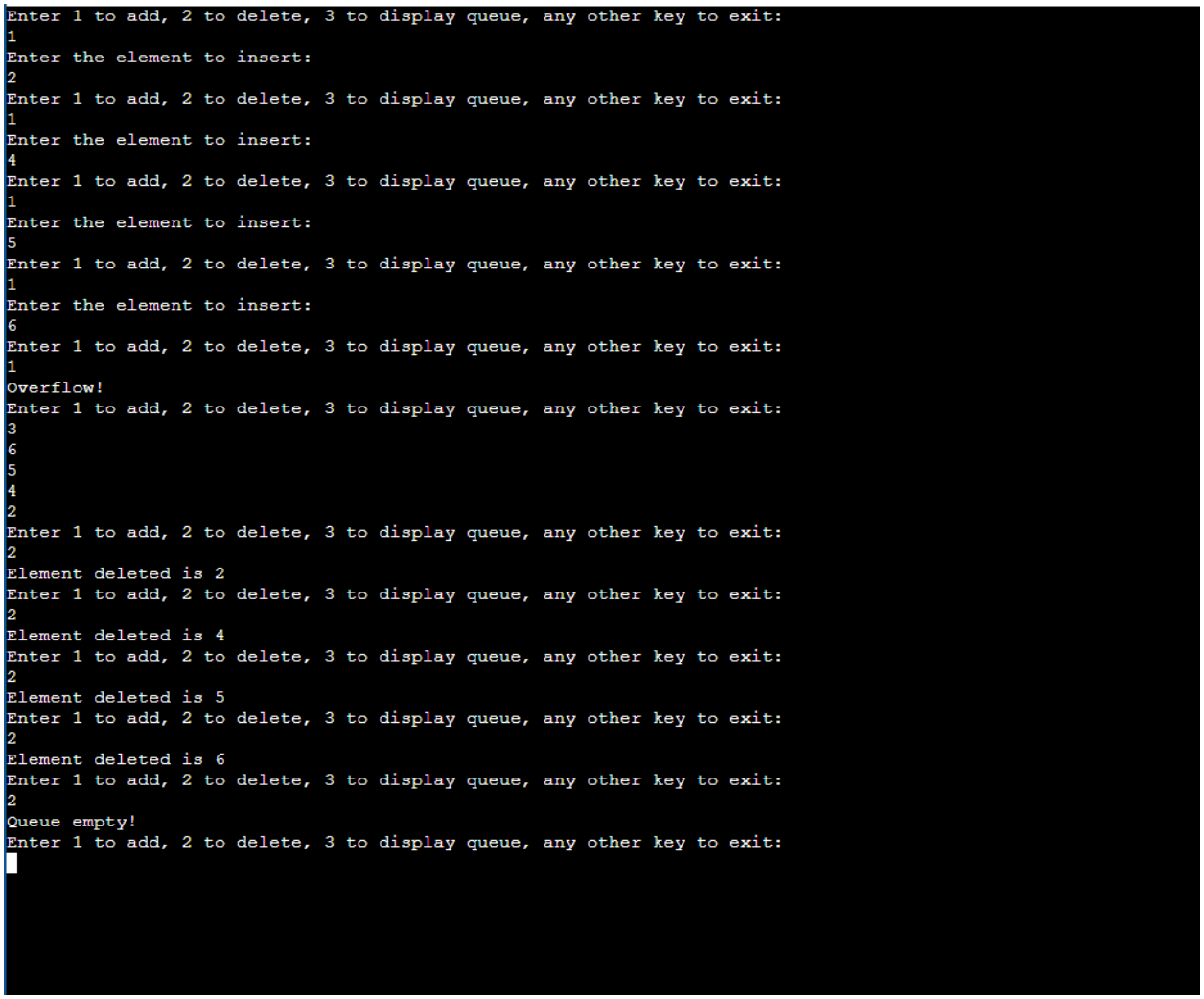
```
        val = q[FRONT];
        FRONT++;
        printf("Element deleted is %d\n", val);
    }
}

void display() {
    int i;
    for (i = REAR; i >= FRONT; i--) {
        printf("%d\n", q[i]);
    }
}

int main() {
    int choice;
    while (1) {
        printf("Enter 1 to add, 2 to delete, 3 to display queue, any other key to
exit:\n");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                enq();
                break;
            case 2:
                deq();
```

```
            break;
        case 3:
            display();
            break;
        default:
            printf("Invalid key entered\n");
            exit(1);
      }
   }
   return 0;
}
```

OUTPUT:



b) #include <stdio.h>

#include <stdlib.h>

```
#define N 4
int q[N];
int REAR=-1;
int FRONT=-1;
void enq();
void deq();
void display();

void enq(){
   int item;
   printf("enter element to insert:\n");
   scanf("%d",&item);
   if(FRONT==-1 && REAR==-1){
      FRONT=REAR=0;
      q[REAR]=item;
   }
   else if((REAR+1)%N==FRONT){
      printf("queue overflow!\n");
   }
   else{
      REAR=(REAR+1)%N;
      q[REAR]=item;
   }
}
void deq(){
```

```
    if(FRONT==-1 && REAR==-1){
        printf("empty queue!\n");
    }
    else if(FRONT==REAR){
        printf("the deleted element is: %d\n",q[FRONT]);
        FRONT=REAR=-1;
    }
    else{
        printf("deleted element:%d\n",q[FRONT]);
        FRONT=(FRONT+1)%N;
    }
}
void display(){
    int i;
    if (FRONT == -1 && REAR == -1) {
        printf("Queue is empty\n");
    }
    else {
        printf("Queue elements: ");
        i = FRONT;
        while (i != REAR) {
            printf("%d ", q[i]);
            i = (i + 1) % N;
        }
        printf("%d", q[REAR]); // Print the last element
```

```
    }
    printf("\n");
}

void main(){
    int choice;
    while(1){
    printf("enter 1. insert 2. delete 3. display\n");
    scanf("%d",&choice);
    switch(choice){
        case 1: enq();
            break;
        case 2: deq();
            break;
        case 3: display();
            break;
        default: printf("invalid entry\n");
            exit(0);
    }
    }
}
```

OUTPUT:

P.T.O

```
enter 1. insert 2. delete 3. display
1
enter element to insert:
2
enter 1. insert 2. delete 3. display
1
enter element to insert:
6
enter 1. insert 2. delete 3. display
1
enter element to insert:
7
enter 1. insert 2. delete 3. display
1
enter element to insert:
9
enter 1. insert 2. delete 3. display
1
enter element to insert:
8
queue overflow!
enter 1. insert 2. delete 3. display
3
Queue elements: 2 6 7 9
enter 1. insert 2. delete 3. display
2
deleted element:2
enter 1. insert 2. delete 3. display
2
deleted element:6
enter 1. insert 2. delete 3. display
2
deleted element:7
enter 1. insert 2. delete 3. display
2
the deleted element is: 9
enter 1. insert 2. delete 3. display
2
empty queue!
enter 1. insert 2. delete 3. display
```