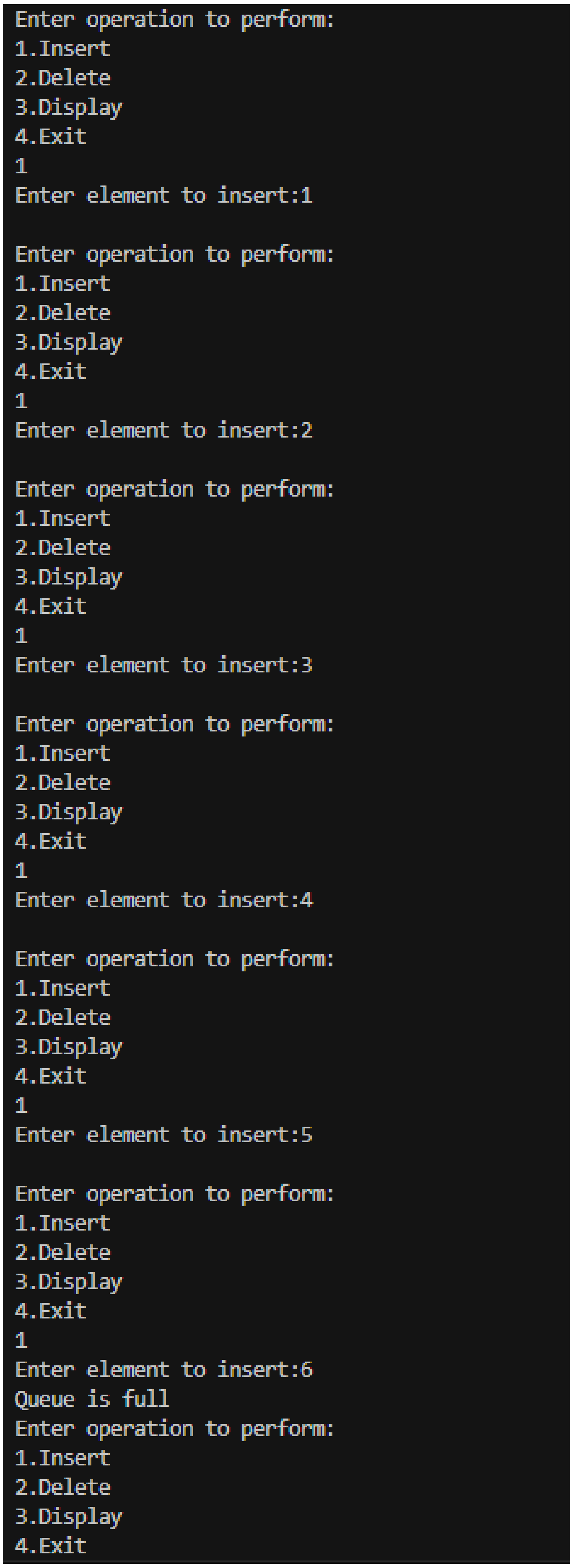
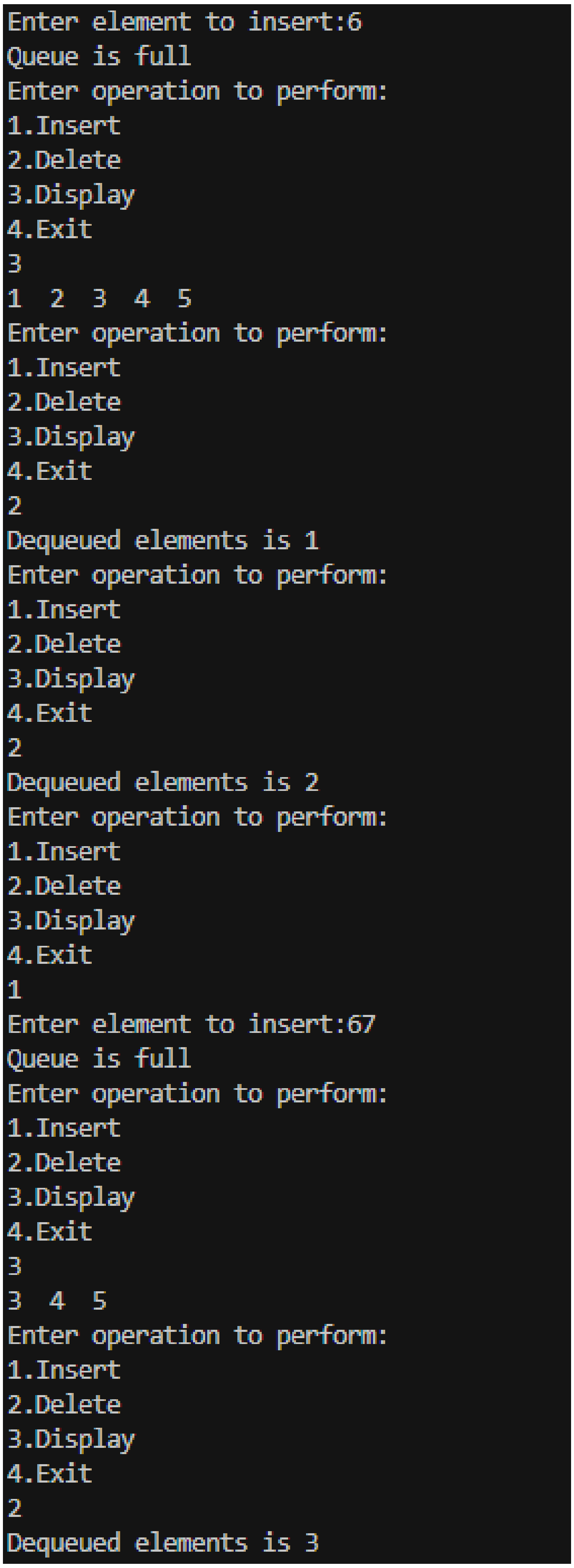
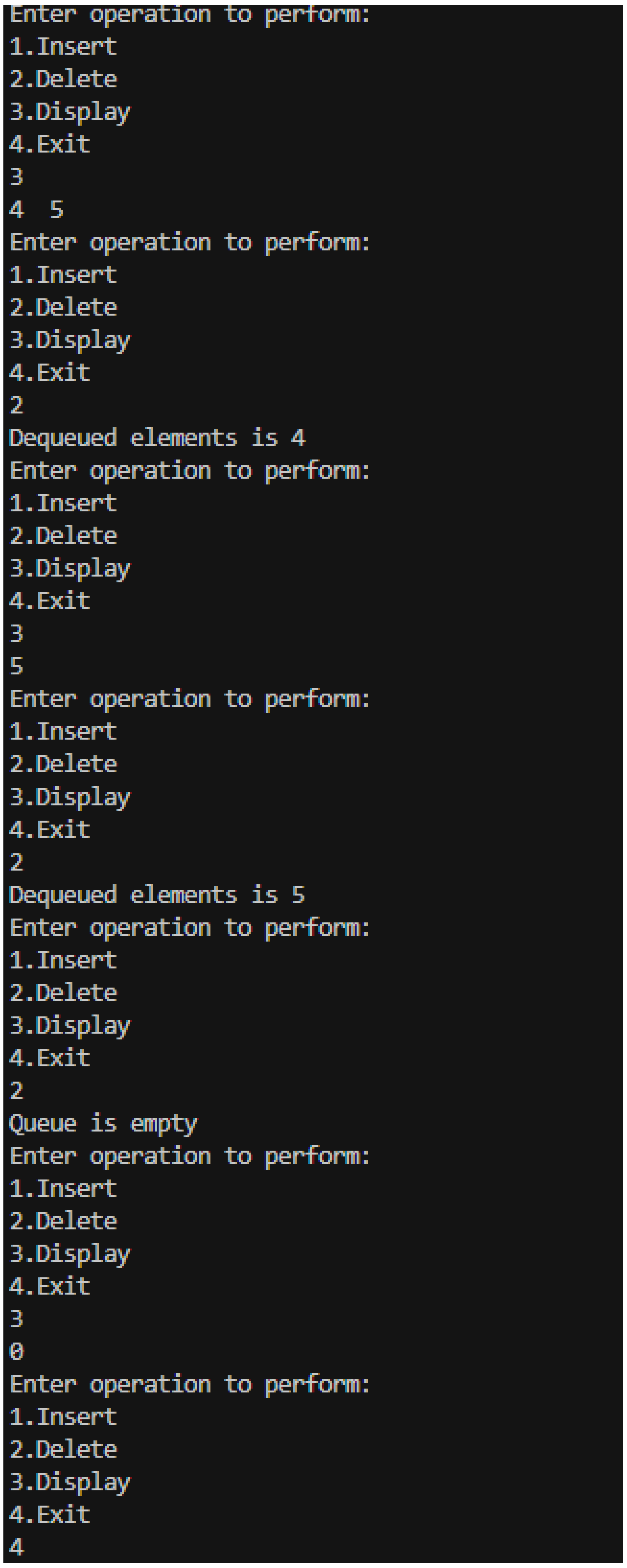
OUTPUT:



```
Enter operation to perform:
1.Insert
2.Delete
3.Display
4.Exit
1
Enter element to insert:1

Enter operation to perform:
1.Insert
2.Delete
3.Display
4.Exit
1
Enter element to insert:2

Enter operation to perform:
1.Insert
2.Delete
3.Display
4.Exit
1
Enter element to insert:3

Enter operation to perform:
1.Insert
2.Delete
3.Display
4.Exit
1
Enter element to insert:4

Enter operation to perform:
1.Insert
2.Delete
3.Display
4.Exit
1
Enter element to insert:5

Enter operation to perform:
1.Insert
2.Delete
3.Display
4.Exit
1
Enter element to insert:6
Queue is full
Enter operation to perform:
1.Insert
2.Delete
3.Display
4.Exit
```

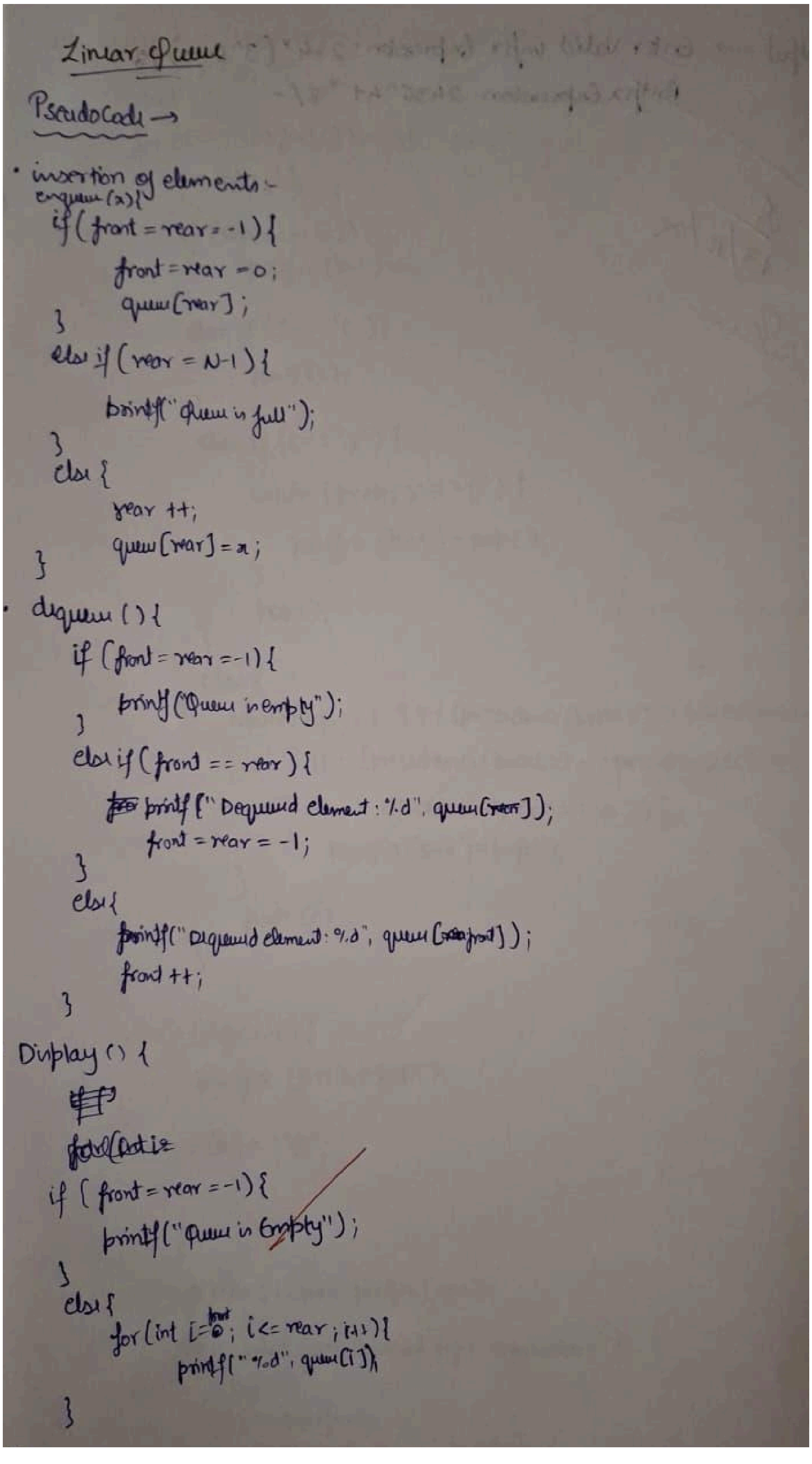


```
Enter element to insert:6
Queue is full
Enter operation to perform:
1.Insert
2.Delete
3.Display
4.Exit
3
1  2  3  4  5
Enter operation to perform:
1.Insert
2.Delete
3.Display
4.Exit
2
Dequeued elements is 1
Enter operation to perform:
1.Insert
2.Delete
3.Display
4.Exit
2
Dequeued elements is 2
Enter operation to perform:
1.Insert
2.Delete
3.Display
4.Exit
1
Enter element to insert:67
Queue is full
Enter operation to perform:
1.Insert
2.Delete
3.Display
4.Exit
3
3  4  5
Enter operation to perform:
1.Insert
2.Delete
3.Display
4.Exit
2
Dequeued elements is 3
```



```
Enter operation to perform:
1.Insert
2.Delete
3.Display
4.Exit
3
4  5
Enter operation to perform:
1.Insert
2.Delete
3.Display
4.Exit
2
Dequeued elements is 4
Enter operation to perform:
1.Insert
2.Delete
3.Display
4.Exit
3
5
Enter operation to perform:
1.Insert
2.Delete
3.Display
4.Exit
2
Dequeued elements is 5
Enter operation to perform:
1.Insert
2.Delete
3.Display
4.Exit
2
Queue is empty
Enter operation to perform:
1.Insert
2.Delete
3.Display
4.Exit
3
0
Enter operation to perform:
1.Insert
2.Delete
3.Display
4.Exit
4
```

OBSERVATION:



Linear Queue

PseudoCode →

```
• insertion of elements -
enqueue(x){
  if(front = rear = -1){
      front = rear = 0;
      queue[rear];
  }
  else if (rear = N-1){
      printf("Queue is full");
  }
  else {
      rear ++;
      queue[rear] = x;
  }
• dequeue(){
    if (front = rear = -1){
        printf("Queue is empty");
    }
    else if (front == rear){
        ~~[illegible]~~ printf("Dequeued element : %d", queue[~~rear~~]);
        front = rear = -1;
    }
    else{
        printf("Dequeued element: %d", queue[~~rear~~front]);
        front ++;
    }
Display () {
    ~~[illegible]~~
    ~~for(int i=~~
   if ( front = rear = -1){
       printf("Queue is Empty");
   }
   else{
       for (int i=~~0~~front; i<= rear; i++){
           printf("%d", queue[i]);

   }
```

Linear Queue

code→

```
#include <stdio.h>

define N10

int queue[N];
int front =-1;
int rear =-1;

int enqueue(){
    int x;
    printf("enter element to insert:");
    scanf("%d", &x);
    if (rear==N-1){
        printf("Queue is full");
    }
    else if(front==-1 && rear==-1){
        front= rear=0;
        queue[rear]=x;
    }
    else {
        rear++;
        queue[rear]=x;
    }
}
int dequeue(){
    if(front==-1 && rear==-1){
        printf("Queue is empty");
    }
    else if(front == rear){
        printf("Dequeued element is %d", queue[rear]);
        front = rear =-1;
    }
    else {
        printf("Dequeued element is %d", queue[front]);
        front++;
}   }
```

```
void display(){
        if (front == rear == -1){
              printf("Queue is empty");
        }
        else {
            for (int i=0; i<rear; i++){
              printf("%d  ", queue[i]);
            }
        }
  }

int main(){
     int choice;
     while (choice != 4){
          printf("Enter Operation to perform: \n1. Insert \n2. Delete \n
                3. Display \n4. Exit");
          scanf("%d",&choice);
          switch (choice){
               case 1: Enqueue();
               break;
               case 2: Dequeue();
               break;
               case 3: Display();
               break;
               case 4: break;
               default: printf("Invalid Choice");
          }
     }
}
```