4. List all queue function operations available for manipulation of data elements in c

1. Front - the front variable points to the position of the first element of the queue for quick deletion.

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

2. Rear - the rear variable points to the position of the last element of the queue for quick insertion.

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

3. isEmpty() - This operation is used to check if the queue is empty.

```
int is_empty(){
   if (front==-1 || front>rear){
      return 1;
   }
   return 0;
}
```

4. isFull() - This operation checks to see whether the queue is at full capacity.

```
int isFull(struct queue *q) {
   return (q->rear == MAX_SIZE - 1);
```

5. }peek() - This operation is used to just access the first element of the queue.

```
int peek(struct queue *q) {
   if (isEmpty(q)) {
```

```
        printf("Queue is empty. Cannot peek.\n");
        return -1;
    }

    return q->items[q->front];
}
```

6. size() - This operation is used to return the size of the queue.

```
    int count(){
    int count = 0;
    if (rear == -1)
        printf("Queue is empty\n");

    else if(front != -1 && rear != -1){
        for (int i = front; i <= rear; i++){
            count++;
        }
    }
    return count;
}
```

7. enqueue() - This method is used to insert elements into the queue.

```
void enqueue(int val){
    if (rear == SIZE-1)
    {
```

```
        printf("\n Queue is Full \n");
    }
    else{
        if (front == -1){
            front = 0;
    }
    rear++;
    queue[rear] = val;
    }
}
```

8. dequeue() - This method is used to delete elements from the queue.

```
void dequeue(){
    if (front==-1 || front>rear){
        printf("Queue is empty\n");
        return;
    }
    front++;
}
```