Outline

Queues (Continued)
- Examples on Queue Applications
- Priority Queues
  - Ascending Priority Queues
  - Descending Priority Queues

Ex: Assume that we have a queue of integer numbers. Write a function, QueueSearch, to search a given key element in the queue until the search key is found. Once the search key is found, the function returns its position in the queue, otherwise returns -1 to indicate that the searched key is not in the queue.

Assume that the Queue contains integer elements and has the following structure:

```c
typedef struct{
    int front;
    int rear;
    int items[MAXQUEUE]; /* Assume that MAXQUEUE is defined*/
}QUEUE;
```

Then the following function can be written:

```c
int QueueSearch(QUEUE *qptr, int searchkey) {
    int pos = -1, f;
    f = qptr->front;
    while(qptr->front != qptr->rear){
        if(qptr->front == MAXQUEUE-1)
            qptr->front = 0;
        else
            qptr->front ++;
        if(qptr->items[qptr->front] == searchkey){
            pos = qptr->front;
            qptr->front = f;
            return pos;
        }
    }
    qptr->front = f;
    return pos;
}
```
Ex: Write a function, QueueCopyReverse, to copy the integer elements of Queue 1 to Queue 2 in reverse order. Assume that there is enough space in Queue 2 for copying and the size of both of the Queues is MAXQUEUE.

```c
void QueueCopyReverse(QUEUE *qptr1, QUEUE *qptr2)
{
int r,x;
r=qptr1->rear;
do{
    x = qptr1->items[qptr1->rear];
    insert(qptr2,x);
    if(qptr1->rear ==0)
        qptr1->rear = MAXQUEUE-1;
    else
        qptr1->rear --;
}while(qptr1->rear != qptr1->front);
qptr1->rear = r;
}

void insert(QUEUE *qptr, int x)
{
   if(qptr->rear == MAXQUEUE-1)
       qptr->rear=0;
   else
       qptr->rear++;

   if(qptr->rear == qptr->front){
       printf("Queue overflow");
       exit(1);
   }
   qptr->items[qptr->rear]=x;
}
```

**PRIORITY QUEUES**

The priority queue is a data structure in which intrinsic ordering of the elements determines the results of its basic operations.

An *ascending priority queue* is a collection of items into which items can be inserted arbitrarily and from which only the smallest item can be removed. On the other hand a *descending priority queue* allows only the largest item to be removed.

- **Insertion**
  - The insertion in Priority queues is **the same as** in non-priority queues.

- **Deletion**
  - Deletion requires a search for the element of highest priority and deletes the element with highest priority. The following methods can be used for deletion/removal from a given Priority Queue:
    - An empty indicator replaces deleted elements.
    - **After each deletion elements can be moved up in the array decrementing the rear.**
    - The array in the queue can be maintained as an ordered circular array
Queue data type of Priority Queue is the same as the Non-priority Queue.

```c
#define  MAXQUEUE  10 /* size of the queue items*/
typedef struct queue{
    int front, rear;
    int items[MAXQUEUE];
}QUEUE;
```

PriQremove Operation using removing the element with highest priority and shifting the elements up in the array and decrementing rear. Consider Ascending Priority Queue.

```c
int PriQremove(QUEUE *qptr)
{
    int smallest, loc, f, i;
    f=qptr->front;
    if(qptr->front == qptr->rear){
        printf("Queue underflow");
        exit(1);
    }
    smallest = qptr->items[(qptr->front+1)%MAXQUEUE];
    loc = (qptr->front+1)%MAXQUEUE;
    (qptr->front++)%MAXQUEUE; /* Circular increment*/
    while(qptr->front != qptr->rear){
        if(qptr->items[(qptr->front+1)%MAXQUEUE] < smallest){
            smallest = qptr->items[(qptr->front+1)%MAXQUEUE];
            loc = (qptr->front+1)%MAXQUEUE;
        }
        qptr->front = (qptr->front++)%MAXQUEUE; /* Circular inc.*/
    }
    while(loc != qptr->rear){
        qptr->items[loc] = qptr->items[(loc+1)%MAXQUEUE];
        (loc++)%MAXQUEUE;
    }
    qptr->front=f;
    if(qptr->rear == 0) /*Decrement rear after removing one item*/
        qptr->rear = MAXQUEUE -1;
    else
        qptr->rear--;
    return smallest;
}
```

Insert Operation of Priority Queue is the same as the insert of the non-priority queues.

```c
void insert(struct queue *qptr, int x)
{
    qptr->rear = (qptr->rear++)%MAXQUEUE; /*Circular increment*/
    if(qptr->rear == qptr->front){
        printf("Queue overflow");
        exit(1);
    }
    qptr->items[qptr->rear]=x;
}
```