Assignment 4 – Queues and Lists

Write pseudo-code not Java for problems requiring code. You are responsible for the appropriate level of detail.

1. Develop an ADT specification for a priority queue. A priority queue is like a FIFO queue except that items are ordered by some priority setting instead of time. In fact, you may think of a FIFO queue as a priority queue in which the time stamp is used to define priority.

```
ADT Priority Queue
        Data
                Queue contains object with value, priority value, and pointer
        Methods
                Push(obj)
                        input: queue is initialized
                        precondition:
                        process: object is placed at its place in the queue. The pointer at the index before
points at the new item in the queue. The object pointer points to next index in queue
                        postcondition: queue is ordered
                        output:
                Pop()
                        input:
                        precondition: there is something in the queue
                        process: the item at the top of the queue is removed from the queue. The item
before the last item has its pointer changed to null
                        postcondition: the queue is ordered
                        output: the object at the top of the queue
                getSize()
                        input:
                        precondition: there is something in the queue
                        process: loop over items in the queue until null, counting each iteration
                        postcondition:
                        output: the size of the queue
End ADT Priority Queue
```

2. Write an algorithm to reverse a singly linked list, so that the last element become the first and so on. Do NOT use Deletion - rearrange the pointers.

```
def reverseList(List list) {
   lastItem = null;  // initialize lastItem to null to trigger first if statement
   for item in list {
      if(lastItem == null) {
         item.pointer = null; // point the first item to null
      } else if(item.pointer == null){
         list.head.pointer = item; // point the head to the last item
      } else {
         item.pointer = lastItem; // point the item to the element before it
```

```
        }
        lastItem = item; // update value of lastItem
    }
}
```

3. What is the average number of nodes accessed in search for a particular element in an unordered list? In an ordered list? In an unordered array? In an ordered array? Note that a list could be implemented as a linked structure or within an array.

Length of list = N

Unordered list: Should be N/2.

Ordered list: Depends on the search algorithm. For example, if you search an ordered list in order, it will be the same as an unordered list (N/2). But if you utilize binary search, your average search time will be decreased to lg(N)

Unordered array: Should be N/2 again.

Ordered array: Should be the same as an ordered list, where it depends on the search algorithm

4. Write a routine to interchange the *m*th and *n*th elements of a singly-linked list. You must rearrange the pointers, not simply swap the contents.

```
def getPreceedingItem(obj i) { // not that efficient, but it works
    preceedingItem = list.head; // if you hit on the first item the preceeding item is the head of the list
    for item in list {
        if(item == i) {
            return preceedingItem; // return the item before
        }
        preceedingItem = item; // reset the item before
    }
    return false; // return false if item i is not in list

def swap(m, n) {
    preceedingM = getPreceedingItem(m);
    preceedingN = getPreceedingItem(n);

    afterM = m.pointer; // get value after m
    afterN = n.pointer; // get value after n (not 100% necessary, but makes for more readable code)

    preceedingM.pointer = n;
    n.pointer = afterM;
    preceedingN.pointer = m;
    m.pointer = afterN;
}
```