1. (10 points) Given a priority queue with the interface:

   void Insert(Item i, int priority);
   Item DeleteMax();

Describe a how to implement a stack with the interface:

   void Push(Item i);
   Item Pop();

You may use $O(1)$ extra space in the stack, but the data must be stored in the priority queue.

2. A 2-3 tree is a B-tree in which the branching factor is 3. That is, there at most two items per node and each node has either two or three children. Answer the following questions giving exact functions of $n$ (i.e. do not use $O$-notation).

(a) (5 points) How many keys (not items) must be examined in the worst case in searching a 2-3 tree containing $n$ items if every node has three children?

(b) (5 points) How many keys (not items) must be examined in the worst case in searching a 2-3 tree containing $n$ items if every node has two children?
3. Suppose that you were provided with code for a Priority Queue in class **PriorityQueue**. The items stored in the priority queue are objects of class **PQItem**. Here is the what the class looks like:

```java
public class PriorityQueue{

    PriorityQueue()  // initializes an empty priority queue

    public void insert(PQItem i)  // inserts i into the PQ

    public PQItem remove()  // removes the PQItem with
    // the maximum priority

}
```

(a) *(10 points)* Write a short method below which takes in an array of PQItems and sorts them according to their priority using the Priority Queue.

(b) *(5 points)* Suppose that the Priority Queue was implemented so that each operation (insert and delete) take only $O(\log \log n)$ time. What would the total running time for your sorting algorithm be?
4. (15 points) Consider the following outline of quicksort:

```plaintext
procedure QuickSort(List);
begin
    if (list has more than one item) then
        begin
            Choose a pivot element from the list;
            Partition list into two lists, L and R, using the chosen pivot.
            Sort L using QuickSort(L)
            Sort R using QuickSort(R)
        end
    else (Do nothing- list already sorted)
end
```

(a) What is the worst-case choice for a pivot?

(b) What is the best-case choice for a pivot?

(c) The median of a set of \( n \) numbers is a number \( x \) such that at least \( \lfloor n/2 \rfloor \) numbers are at most \( x \) and at least \( \lceil n/2 \rceil \) are at least \( x \). In other words, if the numbers were to be sorted, the median would be in the middle of the list.

Suppose that someone gives you a method \texttt{FindMedian} to find the median of \( n \) numbers in \( O(n) \) time.

How would you use \texttt{FindMedian} to improve the Quicksort method outlined above? Express your answer by filling in the outline of Quicksort written on the next page.
procedure \textsc{QuickSort}(List); 
begin 
    if (list has more than one item) then 
        begin 
            Sort L using \textsc{QuickSort}(L) 
            Sort R using \textsc{QuickSort}(R) 
        end 
    else (Do nothing- list already sorted) 
        end 
end

(d) Write a recurrence relation for the worst-case running time for your new version of Quicksort.

(e) What is the worst-case running time for the new version of quicksort? You should express your answer using $O$-notation.