LEBANESE AMERICAN UNIVERSITY

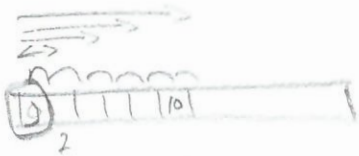
School of Arts and Science

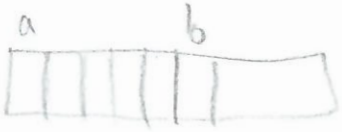
Department of Computer Science and Mathematics

CSC 310: Algorithms and Data Structures

Fall 2014

Lab III

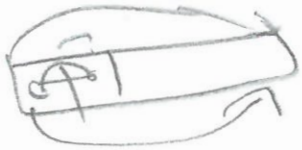




Problem 1: [Heap - problem1.in]

Implement a heap data structure class with the following methods:

- **left**: returns the index of the left child
- **right**: returns the index of the right child
- **parent**: returns the index of the parent
- **max**: returns the max value in the heap
- **extractMax**: returns the element with the max value after removing it from the heap
- **increaseKey**: update a key within a heap
- **maxHeapify**: arranges elements to form a heap
- **print**: prints the heap



Your heap implementation should be done using an array.

Hint 1: The insert method uses the increaseKey method.
Hint 2: The extractMax method uses the maxHeapify method.

Your program will be tested for a number of test cases. The first line of input is an integer *T* representing the number of test cases. *T* lines follow, each beginning with an integer *N* representing the number of elements in the heap. *N* integers follow representing the elements you need to insert into the heap.

After inserting all the elements, you should print the heap, then extract the max value using the extractMax method and print it. Then, you should print the heap again after extracting the max value.

Sample Input:

```
2
10 4 1 3 2 16 9 10 14 8 7
7 12 31 9 10 22 7 1
```

Sample Output:

```
16 14 10 8 7 3 9 1 4 2
16
14 8 10 4 7 3 9 1 2
31 22 9 10 12 7 1
31
```

22 12 9 10 1 7

Problem 2: [Heap Sort - problem2.in]

Sort the numbers using heap sort.

The first line of input is an integer *T* representing the number of test cases. Each test case starts with an integer *N* which is the number of elements to sort then followed by the elements.

Sample Input:	**Sample Output:**
2	
7 5 8 12 6 9 4 1	1 4 5 6 8 9 12
4 22 23 45 20	20 22 23 45

Problem 3: [Quick Sort - problem3.in]

Sort the numbers using quick sort. Choose the pivot to be the first element in the array.

The first line of input is an integer *T* representing the number of test cases. Each test case starts with an integer *N* which is the number of elements to sort then followed by the elements.

Sample Input:	**Sample Output:**
2	
7 5 8 12 6 9 4 1	1 4 5 6 8 9 12
4 22 23 45 20	20 22 23 45

Problem 4: [Randomized Quick Sort - problem4.in]

Sort the numbers using quick sort. Choose the pivot to be at a random index.

The first line of input is an integer *T* representing the number of test cases. Each test case starts with an integer *N* which is the number of elements to sort then followed by the elements.

Sample Input:	**Sample Output:**
2	
7 5 8 12 6 9 4 1	1 4 5 6 8 9 12
4 22 23 45 20	20 22 23 45

Problem 5 - Bonus: [Fast Sort - problem5.in]

Sort the numbers using a sorting algorithm the will be faster the Java's Arrays.sort method.

The first line of input is an integer *T* representing the number of test cases. Each test case starts with an integer *N* which is the number of elements to sort then followed by the elements.

Sample Input:

2
7 5 8 12 6 9 4 1
4 22 23 45 20

Sample Output:

1 4 5 6 8 9 12
20 22 23 45