ESO207A: Data Structures and Algorithms

Practice Set *2: Heaps and Stacks* –

1. A Max-Priority Queue supports the operation INCREASE-KEY(A, i, v) that increases $A[i]$ to v. Can you write the operation DECREASE-KEY(A, i, v) that decreases $A[i]$ to v to work in worst-case $O(\log n)$ time.

2. Design a data structure that supports the following operations: INSERT, MEDIAN and EXTRACT-MEDIAN of a dynamic set. Each of the operations should run in time $O(\log n)$ worst-case time. (*Hint:* Use two heaps).
Can you extend the above data structure to additionally support the operations MAXIMUM and EXTRACT-MAX in worst-case $O(\log n)$ time.

3. You are given an arithmetic expression that can contain three types of brackets, left and right parenthesis, left and right braces and left and right square brackets, namely, '(' and ')', '{' and '}' and '[' and ']'. Write a program that checks whether the given expression has properly nested and matching parenthesis of all types that occur in it. For example, $\{2 + (3 + [5\%7] * 10)/3\} - 9$ is a well-formed expression with matching parenthesis. To test well-formedness in terms of properly nested parenthesis of all types, one can ignore the operators and operands completely and remove them from the expression. This leaves a string consisting only of left and right parenthesis of the three kinds, which can be tested. Examples:

> Ignoring the operands and operators, the following expression is well formed:
> []([{()}]{()}).
>
> Ignoring operators and operands, the following expression is not well-formed:
> [({)}]

4. Given an array of numbers $A[1 \ldots n]$, for each i, $1 \leq i \leq n$, $g(i)$ is defined as the nearest index $j \geq i$ such that $A[j] > A[i]$. Let $g(n) = \infty$ and if $A[i]$ is not smaller than any element in $A[i + 1 \ldots n]$, then we set $g(i) = \infty$. (You may assume that the numbers in the array are all distinct, if needed, though this is not necessary). For example, consider the array $A = \{10, 7, 6, 8, 9, 12, 1\}$. Then, $g(1) = 6$, $g(2) = g(3) = 4$, $g(4) = 5$, $g(5) = 6$, $g(6) = g(7) = \infty$. Give a linear-time algorithm to compute $g[1 \ldots n]$.

5. You are playing a game that allows the following moves. Given an initial string of characters s and empty strings t and u initially. The moves are :

- Remove a character from the front of s and append it to the end of t.
- Remove a character from the end of t and append it to the end of u.

The game ends when s is empty and t is empty and the answer of the game is u. Design a program that makes u to be lexicographically the smallest string possible. For example, suppose the input is `cab`. Then, consider the following moves.

s	t	u
cab		
ab	c	
b	ca	
b	c	a
	cb	a
	c	ab
		abc

Clearly, `abc` is lexicographically the smallest string possible for u.

Example 2. Consider the input string $s =$ `acdb`.

s	t	u
acdb		
cdb	a	
cdb		a
db	c	a
b	cd	a
	cdb	a
	cd	ab
	c	abd
		abdc

The smallest string in lexicographic order possible is `abdc`.