CS 6103D Software Systems Laboratory

PROBLEM 1C

The objective is to learn the following:

- implementation of binary search tree using pointers
- implementation of stack using pointers (as a singly linked list)
- implementation of priority queue using heap

Submission date: on or before 12.09.2022 Monday 11.59 PM
Submission: a single file named as per the following format

- Submit as a single .tar file
- The name of this file must be *P1C_ < FIRSTNAME > _ < ROLLNO > .tar(eg : P1C_ARUN_M180xxxCS.tar)*

Modify the program developed for problem 1B as follows:

1. Implement the *regList* of each course using a *Binary Search Tree (BST)*. The field *RegList* in a course struct is now a pointer to the root of a *BST*. Each node should contain name, and pointers to its left child, right child and parent. Define functions *insert(x, t)*, to insert name x to the tree t (t is a pointer to the root of the tree), *delete(x, t)* to delete name x from tree t, and *inorderTreeWalk(t)*, a recursive function for doing the inorder traversal of t. Define a function *printRegList(c)*, which given a course code prints the names of students registered in that course in sorted order, by invoking *inorderTreeWalk(t)*. Define each *BST* operation as per the algorithms given in chapter 12 of CLRS(reference given below).
2. Provide a non recursive version of *inorderTreeWalk(t)*. This requires a stack of pointers to tree nodes. Implement this *stack* using an array. Define operations *push(S, x)* to add an element x to the top of the stack S, *pop(S)* to pop out the top most element from stack S and *isEmpty(S)* which returns true if the stack S is empty and false otherwise.
3. Maintain the *waitList* as a *max priority queue*. Each student entering the queue is given a priority value ranging from 1 to *maxLimit* where *maxLimit* is the maximum number of students allowed in the course. Implement this priority queue using a *maxHeap*. Define operations *insert(Q, x)* to insert an element x to the priority queue Q, *extract_Max(Q)* to remove and return the element with the highest priority value from Q, *increaseKey(Q, x, k)* to increase the priority value of element x in Q to the new value k (new value is assumed to be at least as large as the current priority value of x). Each heap operation is to be implemented as per the algorithms given in section 6.5 of CLRS(reference given below).

- **Reference**: T. H. Cormen, C. E. Lieserson, R. L. Rivest, C. Stein. *Introduction to Algorithms*, PHI Learning, 3rd edition, 2010.