

**Assignment 4**  
Computer Science Tutor 2011/2012  
Due 16-05 2012

A binary search tree  $T$  is a data-structure that can be used to maintain an ordered set of data elements that typically supports the following dynamic-set operations:

**Isempty()**

- equal to true, if the tree  $T$  is empty
- equal to false, if the tree  $T$  contains at least one element

**Insert(key)**

- if the data element key is not already stored in  $T$ , it will be stored in  $T$
- if the data element key is already stored in  $T$ , nothing happens

**Delete(key)**

- if the data element key is stored in  $T$ , it will be deleted from  $T$
- if the data element key is not stored in  $T$ , nothing happens

**OrderedList()**

- an ordered list of the elements stored in  $T$  will be printed

**Query(key)**

- equal to true, if the data element key is stored in  $T$
- equal to false, if the data element key is not stored in  $T$

**Max()**

- if the tree  $T$  is not empty, equal to the largest element stored in  $T$
- if the tree  $T$  is empty, equal to -1

**Min()**

- if the tree  $T$  is not empty, equal to the smallest element stored in  $T$
- if the tree  $T$  is empty, equal to -1

**Successor(key)**

- returns the smallest element stored in  $T$  that is bigger than the given data element  $\langle \text{key} \rangle$
- if this element does not exist, it returns -1

**Predecessor(key)**

- returns the largest element stored in  $T$  that is smaller than the given data element  $\langle \text{key} \rangle$
- if this element does not exist, it returns -1

In this assignment you are asked to implement a binary tree that stores strictly positive integers. A user should be able to issue commands at the command line that have the following forms and results:

- 'e'** the program will respond with 'T is empty', or 'T is not empty', if the tree T is empty, not empty, respectively.
- 'i <number>'** where <number> is a strictly positive integer; resulting in the <number> being inserted in T.
- 'd <number>'** where <number> is a strictly positive integer; resulting in the <number> being deleted from T.
- 'l'** resulting in a listing of all the elements stored in T ordered from small to large.
- '? <number>'** where <number> is a strictly positive integer; resulting in '<number> is element of T', if <number> is stored in T, and '<number> is not element of T', if <number> is not stored in T.
- 's <number>'** where <number> is a strictly positive integer; resulting in the smallest number in T that is bigger than given <number>. If such a number does not exist in T, the result will be equal to -1.
- 'p <number>'** where <number> is a strictly positive integer; resulting in the largest number in T that is smaller than given <number>. If such a number does not exist in T, the result will be equal to -1.
- 'M'** determines the largest <number> in T, and results in the output '<number> is the largest element in T'.
- 'm'** determines the smallest <number> in T, and results in the output '<number> is the smallest element in T'.
- 'q'** the program stops.

Note: In this assignment you should use an object-oriented approach. You should design and implement a class *CTree* that has all the necessary member functions required for this assignment. Use this class in a program that implements the further requirements of this assignment.