

## Lab 7-1: Binary Search Tree

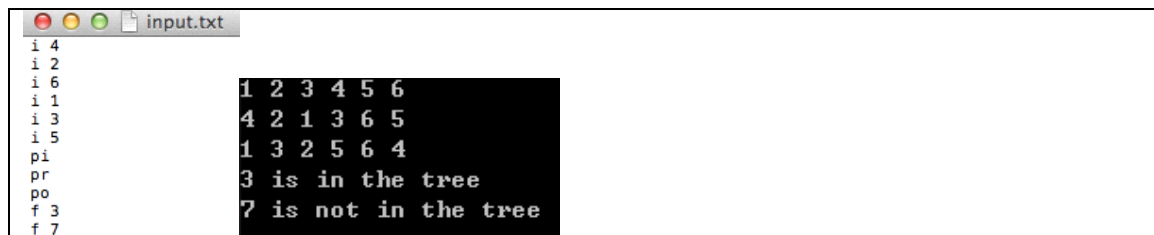
In this lab, we will implement binary search tree ADT with the three main functions, insert, delete, and find. Additionally, we will have three print functions with different ways of traversal.

- **Input**

Obtain a list of operations from the given input file, and execute the given operations in order. A detailed specification of the operations is provided below. Each line represents a single operation. Each operation and the necessary parameters are separated by a space. You may assume that the input values (represented as x below) are any integer.

- **i x**: insert a new key “x” into the binary search tree without duplication. If x already exists in the tree, print an error message.
- **d x**: delete a key “x” in the binary search tree. If x does not exist in the tree, print an error message.
- **f x**: find the given key to check whether the key exists in the tree
- **pi**: print the tree by inorder traversal
- **pr**: print the tree by preorder traversal
- **po**: print the tree by postorder traversal

An input file is shown below.



```
input.txt
i 4
i 2
i 6
i 1
i 3
i 5
pi
pr
po
f 3
f 7

1 2 3 4 5 6
4 2 1 3 6 5
1 3 2 5 6 4
3 is in the tree
7 is not in the tree
```

- **Binary Search Tree ADT**

(1) Data Specification for the objects

```
struct Tree {
    int value;
    Tree *left;
    Tree *right;
};
```

(2) Function specification

- Tree \* insertNode(Tree \*root, int key)
  - Insert a new node with the key value into the tree. If the key already exists in the tree, print an error message.
- Tree \* deleteNode(Tree \*root, int key)
  - delete a node with the given key value from the tree. If the key does not exist in the tree, print an error message.
- Tree \* findNode(Tree \*root, int key)
  - Find the key in the binary search tree.
  - Print “key is in the tree” if the key exists. Otherwise, print “key is not in the tree”.
- void printInorder(Tree \*root)
  - Print the tree by inorder traversal.
- void printPreorder(Tree \*root)
  - Print the tree by preorder traversal.
- void printPostorder(Tree \*root)
  - Print the tree by postorder traversal.

### 3. Program description

- name : p7\_1.c
- input : a list of operations in a file (an input file name is given as a command line argument. See the example in “1. input” on the first page)
- output : the corresponding result in the standard output

## Lab 7-2: Binary max heap

In this lab, we will implement max heap ADT. In a max heap, the keys of parents nodes are always greater than or equal to those of the children nodes. The max key is in the root node. In max heap ADT, we will implement two main functions, insert and deleteMax. Additionally, we will implement a print function, printHeap.

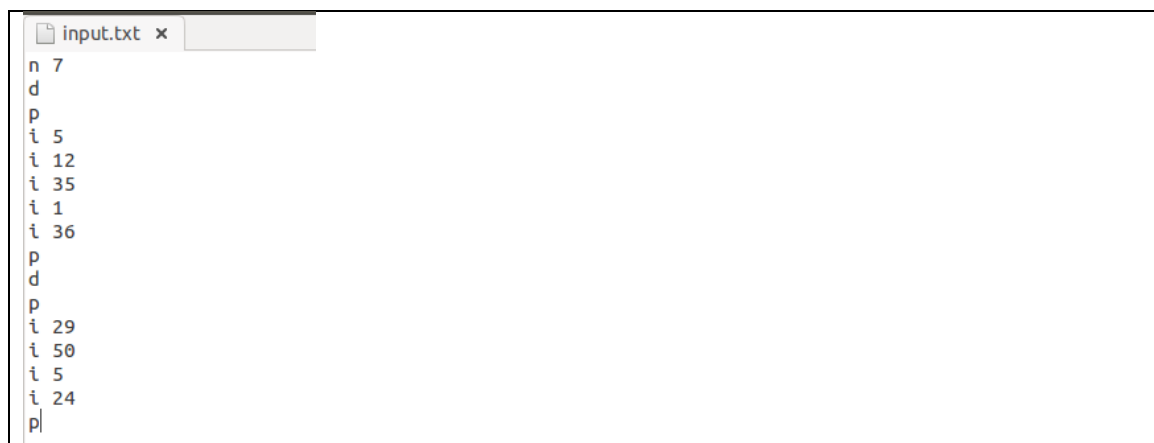
- **Insert:** Insert a new key to the max heap. You should find the right position for the new key to maintain the max heap.
- **DeleteMax:** Delete the max in root node and reconstruct the heap to maintain max heap. If your list does not have any element, just print an error message.
- **PrintHeap:** Print the entire heap. When printing, each level of the max heap should be printed in a line. If your queue is empty, just print an error message.

### 1. Input

Obtain a list of operations from the given input file, and execute the given operations in order. A detailed specification of the operations is provided below. Each line represents a single operation. Each operation and the necessary parameters are separated by a space. You may assume that the input values (represented as x below) are any integer.

- **n x:** create a new heap with the size of x. The number x is the maximum size of the MaxHeap. This operation will always be given in the first line of the operations in your input file
- **i x:** insert a new key "x" into the max heap.
- **d :** delete the max key in the root node
- **p :** print the entire max heap. Print each level of the max heap in a line

An input file is shown below.



```
input.txt x
n 7
d
p
i 5
i 12
i 35
i 1
i 36
p
d
p
i 29
i 50
i 5
i 24
p
```

## 2. MaxHeap ADT

### (1) Data Specification for the objects

```
struct HeapStruct {  
    int Capacity;  
    int Size;  
    ElementType *Elements;  
};
```

### (2) Function specification

- HeapStruct\* CreateHeap(int heapsize);
- void Insert(HeapStruct heap, ElementType value);
- ElementType DeleteMax(HeapStruct heap);
- void PrintHeap(HeapStruct heap);

### 3. Program description

- name : p7.c
- input : a list of operations in a file (an input file name is given as a command line argument. See the example in “1. input” on the first page)
- output : the corresponding result in the standard output

Submit to the course website (<https://portal.hanyang.ac.kr>) your source code and a written report. Your report should include the description of your own implementation.