# CS 202, Spring 2018
## Homework #2 – Binary Search Trees
## Due Date: March 18, 2018

---

## Important Notes

**Please do not start the assignment before reading these notes.**

- Before 23:55, March 18, upload your solutions in a single **ZIP** archive using Moodle submission form. Name the file as `studentID_hw2.zip`.
- Your ZIP archive should contain the following files:
  - `hw2.pdf`, the file containing the answers to Questions 1 and 3,
  - `PbTreeNode.h`, `PbTreeNode.cpp`, `PbBST.h`, `PbBST.cpp`, `analysis.h`, `analysis.cpp`, `main.cpp` files which contain the C++ source codes, and the `Makefile`.
  - Do not forget to put your name, student id, and section number in all of these files. Well comment your implementation. Add a header as in Listing 1 to the beginning of each file:

Listing 1: Header style

```
/**
 * Title: Binary Search Trees
 * Author: Name Surname
 * ID: 21000000
 * Section: 0
 * Assignment: 2
 * Description: description of your code
 */
```

  - Do not put any unnecessary files such as the auxiliary files generated from your favorite IDE. Be careful to avoid using any OS dependent utilities (for example to measure the time).
  - You should prepare the answers of Questions 1 and 3 using a word processor (in other words, do not submit images of handwritten answers).

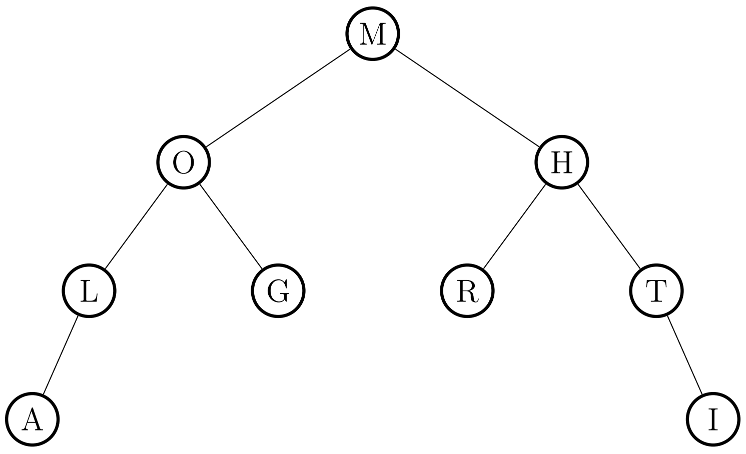
- Although you may use any platform or any operating system to implement your algorithms and obtain your experimental results, your code should work on the **dijkstra** server (`dijkstra.ug.bcc.bilkent.edu.tr`). We will compile and test your programs on that server. Thus, you may lose a significant amount of points if your C++ code does not compile or execute on the dijkstra server.
- This homework will be graded by your TA, Ilkin Safarli. Thus, please contact him directly for any homework related questions.

**Attention:** For this assignment, you are allowed to use the codes given in our textbook and/or our lecture slides. However, you ARE NOT ALLOWED to use any codes from other sources (including the codes given in other textbooks, found on the Internet, belonging to your classmates, etc.). Furthermore, you ARE NOT ALLOWED to use any data structure or algorithm related function from the C++ standard template library (STL).

**Do not forget that plagiarism and cheating will be heavily punished. Please do the homework yourself.**

## Question 1 – 15 points

(a) [*5 points*] What are the preorder, inorder, and postorder traversals of the binary tree below:



(b) [*5 points*] Insert $50, 40, 80, 30, 45, 70, 90, 10, 60, 75, 85$ to an empty Binary Search Tree. Show **only the final tree** after all insertions. Then delete $45, 40, 80, 75, 50$ in given order. Show **only the final tree** after all deletion operations. Use the exact algorithms shown in lectures. Verify your answers by using this visualization tool.

(c) [*5 points*] A binary search tree has a preorder traversal of $P, F, B, M, K, S, R, Z$. What is its postorder traversal? Reconstruct the tree from those traversals and draw it.

## Question 2 – 70 points

(a) [*30 points*] Write a pointer-based implementation of Binary Search Tree named as `PbBST` for maintaining a list of integer keys. Implement `insert`, `deleteKey` and `getHeight` methods for `PbBST` class. Put your code into `PbTreeNode.h`, `PbTreeNode.cpp`, `PbBST.h` and `PbBST.cpp` files. Prototypes of required methods:

```
void insert(int key); // 10 points
void deleteKey(int key); // 15 points
int getHeight(); // 5 points
```

(b) [*10 points*] Write another method for `PbBST` class to return the median of numbers in that BST *in linear time* (linear in the number of items). Your method should have the following prototype:

```
double medianOfBST();
```

(c) [*15 points*] Implement a method for `PbBST` class which prints the keys in a given range (side by side) in $\mathcal{O}(\log n + m)$ time, where $n$ is the number of nodes in the tree and $m$ is the size of the range. You will not get any credit if your implementation runs asymptotically slower. Your method should have the following prototype (assume $a < b$):

```
void rangeSearch(int a, int b);
```

(d) [*15 points*] Height of BST is a very important property which affects the performance of search, delete, and insert operations directly. In this part, you will analyze how the height of BST changes as you insert and delete random number into/from the tree. Write a global function, `void heightAnalysis();` which does the following:

(1) Creates an array of 20000 random numbers and starts inserting them into an empty BST. At each 1000 insertions, outputs the height of the tree.

(2) Shuffles the array created in part d1. Then iterates over it and deletes the numbers from the tree. After each 1000 deletions, outputs the height of the tree.

Add your code to `analysis.cpp` file. When `heightAnalysis` function is called, it needs to produce an output similar to the following one:

Listing 2: Sample output

```
Part f - Analysis of BST height - part 1
----------------------------------------
Tree Size        Tree Height
----------------------------------------
1000                 x
2000                 x
...


Part f - Analysis of BST height - part 2
----------------------------------------
Tree Size        Tree Height
----------------------------------------
19000                x
18000                x
...
```

(e) [*0 points*] Create a `main.cpp` file which does the followings:

- creates an empty BST and inserts the following numbers into it: $\{40, 50, 45, 30, 60, 55, 20, 35, 10, 25\}$
- prints the height of the tree
- deletes 45 and 50 from the tree
- finds the median of numbers by using `medianOfBST` method
- prints numbers between 15 and 53 by using `rangeSearch` method
- calls `heightAnalysis` function

At the end, write a basic Makefile which compiles all your code and creates an executable file named `hw2`. Check out these tutorials for writing a simple make file: tutorial 1, tutorial 2. Please make sure that your `Makefile` works properly on the **dijkstra** server.

## Question 3 – 15 points

After running your programs, you are expected to prepare a single page report [1] about the experimental results that you obtained in Question 2 d. With the help of a spreadsheet program (Microsoft Excel, Matlab or other tools), plot *number of elements in tree* versus *height of tree* after each 1000 insertions and deletions. On the same figure, plot *number*

[1]Please make sure that your report does not exceed the specified page limit too much, otherwise you may lose points

*of elements in tree* versus *theoretical average height of BST*. A sample figure is given in Figure 1 (*these values do not reflect real values*).

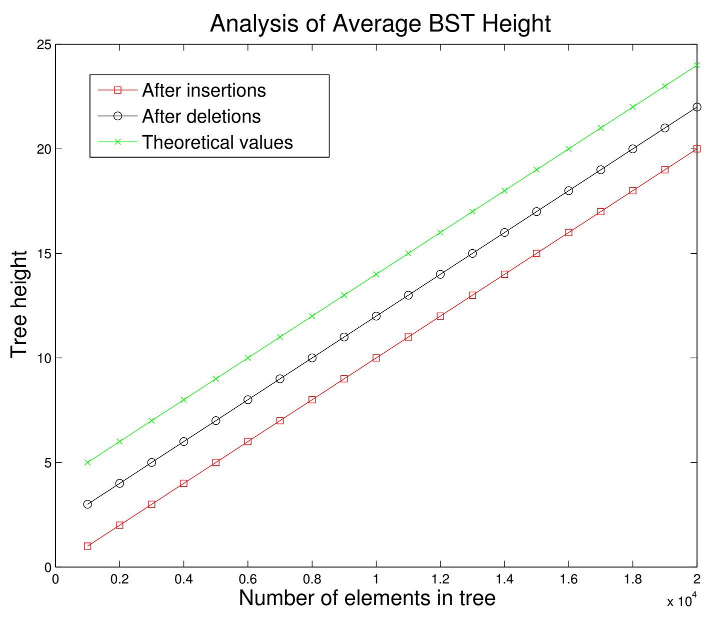


Figure 1: Sample figure

In your report, you need to discuss the following points:

- Do your findings related to average height of BST agree with the theoretical values? State the theoretical value and discuss how close these values are to the values you obtained.
- How would the height of the tree change if you inserted sorted numbers into it instead of randomly generated numbers?