**Module 19 - Ordered operations in BSTs**

**Module Overview**

We examine a symbol-table implementation that combines the flexibility of insertion in linked lists with the efficiency of search in an ordered array. Specifically, using two links per node leads to an efficient symbol-table implementation based on the binary search tree data structure, which qualifies as one of the most fundamental algorithms in computer science.

**Learning Objectives**

1. Ordered operations on BST

### Video on Ordered operations in BSTs

### Fill in the blanks-1

**Fill the following blanks in the rank method of BST**

private int rank(Key key, Node x) {  
       if (x == \_\_\_\_\_\_\_)   return 0;                   **//Blank1**  
       int cmp = key.compareTo(\_\_\_\_\_\_\_\_);              **//Blank2**  
       if (cmp < 0)   return rank(key, \_\_\_\_\_\_\_\_);        **//Blank3**   
       else if (cmp > 0) return 1 + size(x.left) + rank(key, \_\_\_\_\_); **//Blank4**   
       else          return size(\_\_\_\_\_\_);  **//Blank5**

}

**Blank1:**-----------------------------

**Blank2:**-----------------------------

**Blank3:**-----------------------------

**Blank4:-----------------------------**

**Blank5:**-----------------------------

**Module 19 - Ordered operations in BSTs**

**Learning by doing-1**

Give the sequences of nodes examined when the methods in BST are used to compute each of the following quantities for the tree drawn at below.  
a. floor("Q")   
b. select(5)   
c. ceiling("Q")   
d. rank("J")   
e. size("D", "T")   
f. keys("D", "T")

* Answer the above problem in a file name problem-1.txt and add it to the folder ADS-1-practice/m19
* Push the respository to GitHub when your answer is done
* Enter the git commit ID in the blank below

Do not use eval to submit this activity

Top of Form



Bottom of Form

**Assignment-1**

Implement the API using Binary Search >Tree: Implement the following API using Binary Search Tree.  
void put(Key key, Value val)  
Value get(Key k)  
Key min()  
Key max()  
Key floor(Key key)  
Key ceiling(Key key)  
Key select(int k)  
int rank(Key key)  
Iterable keys(Key lo, Key hi)  
Check your program against the given test cases below as a zip file. Submit your Solution(zip file) when all the test cases are passed.

* Download the starter code; the directory structure is similar to the sample-assignment seen in the previous activity
* Add a subfolder m19 for Module 19
* Unzip the starter code into m19 folder. You should see a folder with the name 19.1 BST
* You should write your solution in the file Solution.java
* There are a few lines of code to handle the input testcases
* After you write the code use eval to check if you got all the testcases right
* submit commit ID in the textbox below.