INTERNATIONAL SCHOOL OF ENGINEERING

FACULTY OF ENGINEERING

CHULALONGKORN UNIVERSITY

2190221/2147105 Fundamental Data Structure and Algorithm / DATA STRUC ALG

Second Semester, Midterm Examination March 13, 2023. Time 08:30-11:15

---

Name.............................................Identification no..........................No. in CR58.........

Important

1. This exam paper has 3 questions. There are 8 pages in total including this page. The total mark is 36 (will be scaled down to 30).
2. **All coding questions asked you to write Java code. JUnit is used to test your code. Your code must compile and run in order to get marked.**
3. When the exam finishes, students must stop and remain in their seats until the examiners allow students to leave the exam room.
4. A student must sit at his/her desk for at least 45 minutes.
5. A student who wants to leave the exam room early (must follow (5).) must raise his/her hand and wait for the examiner to check his/her machine. The student must do this in a quiet manner.
6. **Books, lecture notes, written notes, files are allowed.**
7. **All files must be loaded into the machine you use for exam.**
8. A student must not borrow any item from another student in the exam room. If you want to borrow an item, ask the examiner to do it for you.
9. Do not take any part of the exam file out of the exam room. All exam questions are properties of the government of Thailand. Violators of this rule will be prosecuted in a criminal court.
10. A student who violates the rules will be considered as a cheater and will be punished by the following rule:
    - **With implicit evidence or showing intention for cheating, student will receive an F in that subject and will receive an academic suspension for 1 semester.**
    - **With explicit evidence for cheating, student will receive an F in that subject and will receive an academic suspension for 1 year.**

I acknowledge all instructions above. This exam represents **only my own work**. I did not give or receive help on this exam.

Student's signature(........................................)

Name……………………………………..…..ID………..…………………………….CR58……………………

- READ AN ENTIRE QUESTION BEFORE STARTING TO WRITE ANYTHING FOR THAT QUESTION!!!
- In Eclipse, create a workspace called "2022_Datas_{studentID}" where {studentID} is your student id number.
- For each question, create a new Eclipse Project (project name is specified in each question).
- Download 2022_DataS_midterm.zip from assignment in MyCourseville.
- YOU MUST Unzip the file into a folder.!!!!!!!!!!
- Follow the instruction of each question.
- At the end of the exam, leave Eclipse on (minimize the screen).

Question 1 starts next page

Name……………………………………..…..ID……….………………………………CR58……………………

## Q1. Linked List (18 marks)

**WARNING: Q1a) is the most difficult question, you can do other questions first.**

**Create Eclipse project "2022_MDataSt_Q1_{studentID}", where {studentID} is your student ID, and copy all files in folder "Q1_toStudent" to project "src" folder.**

You are given all classes for coding a Linked List that stores characters (one character per node). The characters form a sentence. The list will be used in a typing game:-

- where you type in a word, then the first occurrence of the word (all consecutive nodes that store characters forming that word) is removed from the list.
- The class you must implement is TypingDeadList (that's the name of the typing game we are working on).

```
public class TypingDeadList extends CDLinkedList {
    int score = 0;  // not used in this exam
    DListIterator start = null; // the first position of a word to remove
    DListIterator end = null; // last position of a word to remove
```

- start :-
  - Once a word to remove is given, start marks the node that stores the first character of that word in the list (consider only the first occurrence of the word).
  - Once the word is removed, start becomes null.
- end :-
  - Once a word to remove is given, end marks the node that stores the last character of that word in the list (consider only the first occurrence of the word).
  - Once the word is removed, end becomes null.

Method removeWord(String w) is used to remove a word (assume you already get the word from keyboard) from our TypingDeadList.

```
public void removeWord(String w) throws Exception {
    // remove the first occurrence of w
    // if w is not in the list, do nothing
    // reset start and end to null no matter what
    findWord(w);
    if (start == null)
        return;

    int dec = w.length();
    remove(dec);
}
```

Your task is to write method findWord(w) and remove(dec). Each method is explained as follows:

a) `(8 marks) public void findWord(String w) throws Exception {`

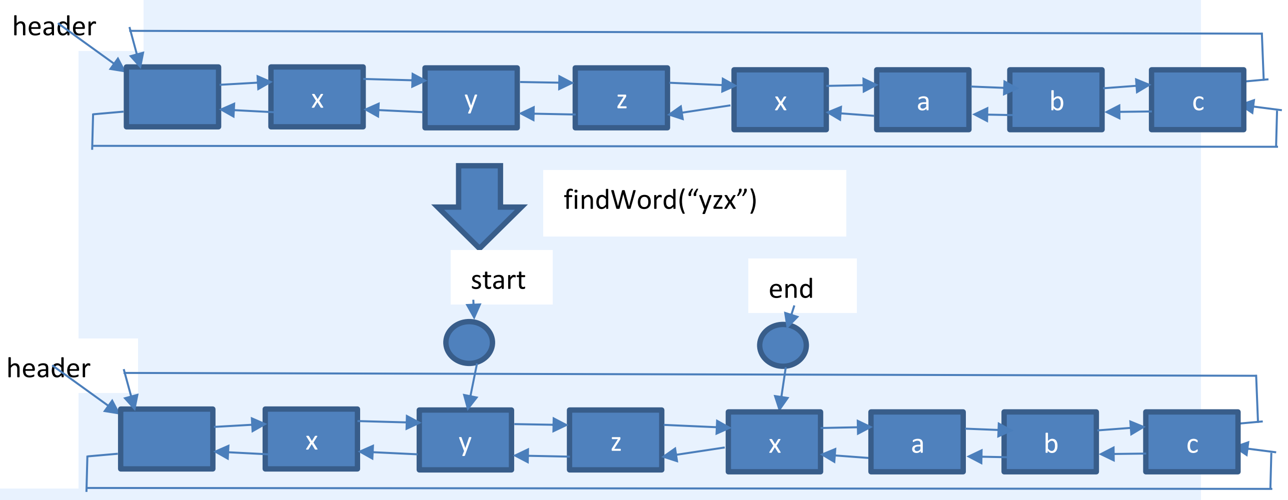
- This method searches the list for the first occurrence of the word w.
- w is assumed never to be an empty string.
- The word, w, cannot overlap the header node.
- If w is not in the list, do nothing.
- Otherwise,
  - update start to mark the position of the first character of w.
  - update end to mark the position of the last character of w.

The test scores are as follows (in file TypingDeadListTest.java):

- testFindWordNotFound1()        1 marks
- testFindWordNotFound2()        1 marks

- testFindWordFound()   6 marks

  Example:



b) (10 marks) `public void remove(int dec) throws Exception {`

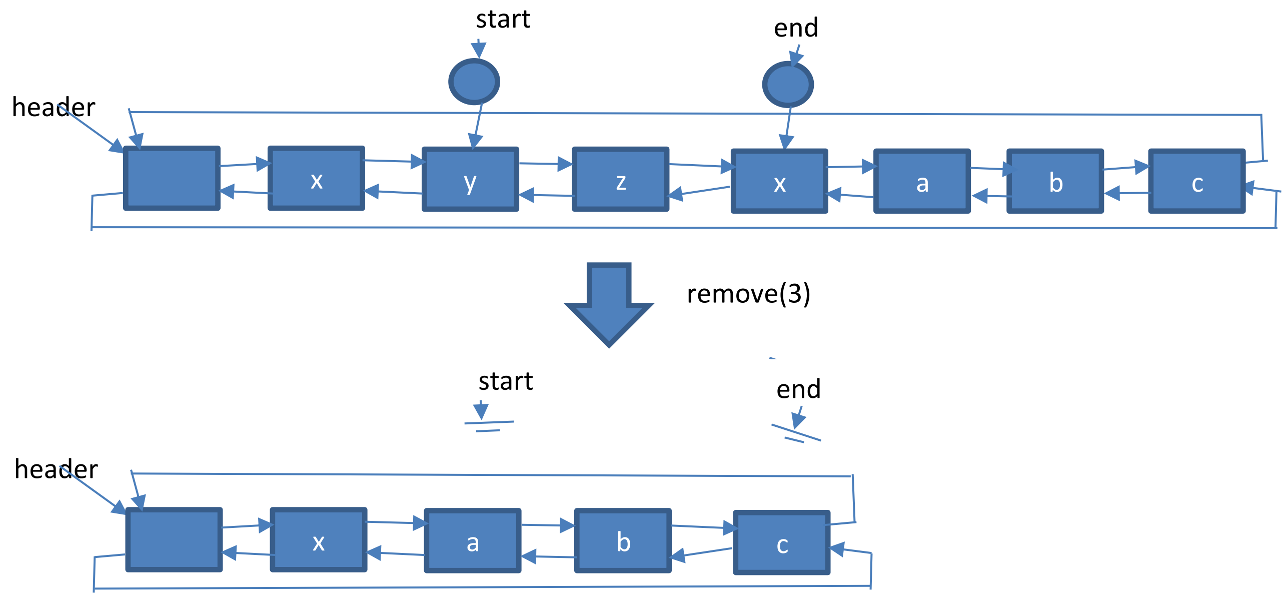
- This method must be the last method in class TypingDeadList. Otherwise, the marking script will not function.
- This method assumes that start and end have already been set.
- It receives the size of the word to be removed.
- If start or end is null, this method does nothing.
- It then removes nodes from start to end (removing includes position start and position end).
- It also updates the list size accordingly.
- Lastly, it resets start and end to null.
- You must not use loop in this method, if you do, you lose 4 marks.

The test scores are as follows (in file TypingDeadListTest.java):

- testRemoveStartOrEndAtHeader()   1 mark
- testRemoveOneValue()   1 mark
- testRemoveAllValue()   2 marks
- testRemoveGeneric()   2 marks
- testNoLoopRemove() (in file TestNoLoop.java)   4 marks
  - If the given path does not work, you must change path in the file to match your file location.

Continue next page

Example: Continued from findWord("yzx") above.



**Q2. Queue (8 marks)**
**Create Eclipse project "2022_MDataSt_Q2_{studentID}", where {studentID} is your student ID, and copy all files in folder "Q2_toStudent" to project "src" folder.**

You are given all classes for coding a queue.
Class QueueArray and QueueLinkedList each has one new method:

```
public int get(int pos) throws Exception {
     //TODO
     //FILL CODE

}
```

This method returns the data at position "pos". (the first position in a queue is 0 (it is position "front" in QueueArray)).

- If the queue is empty, or pos is negative or too large, throw an exception.

Write code for method get(int pos) in both classes.

JUnit is in QueueTest.java

```
•   testGetQueueArrayEmptyQueue()     0.5 marks
•   testGetQueueArrayNegPos()         0.5 marks
•   testGetQueueArrayTooLargePos()    0.5 marks
•   testGetQueueArrayOK()             2.5 marks
•   testGetQueueListEmptyQueue()      0.5 marks
•   testGetQueueListNegPos()          0.5 marks
•   testGetQueueListTooLargePos()     0.5 marks
•   testGetQueueListOK()              2.5 marks
```

Name…………………………………..…..ID……….…………………………….CR58……………………..

## Q3.Tree (10 marks)

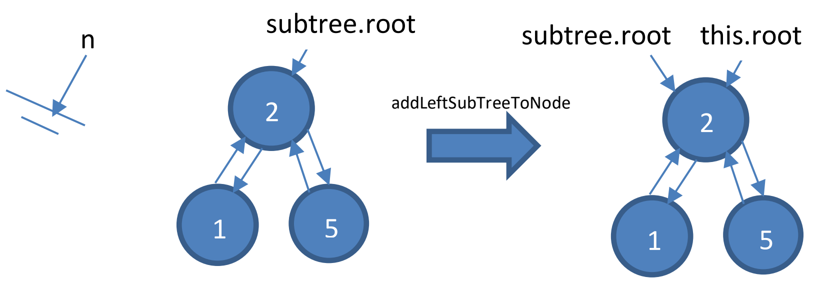
**Create Eclipse project "2022_MDataSt_Q3_{studentID}", where {studentID} is your student ID, and copy all files in folder "Q3_toStudent" to project "src" folder.**

You are given all classes for coding a binary search tree.
In class BST, write method

```
public void addLeftSubTreeToNode(BST subtree, BSTNode n)
```

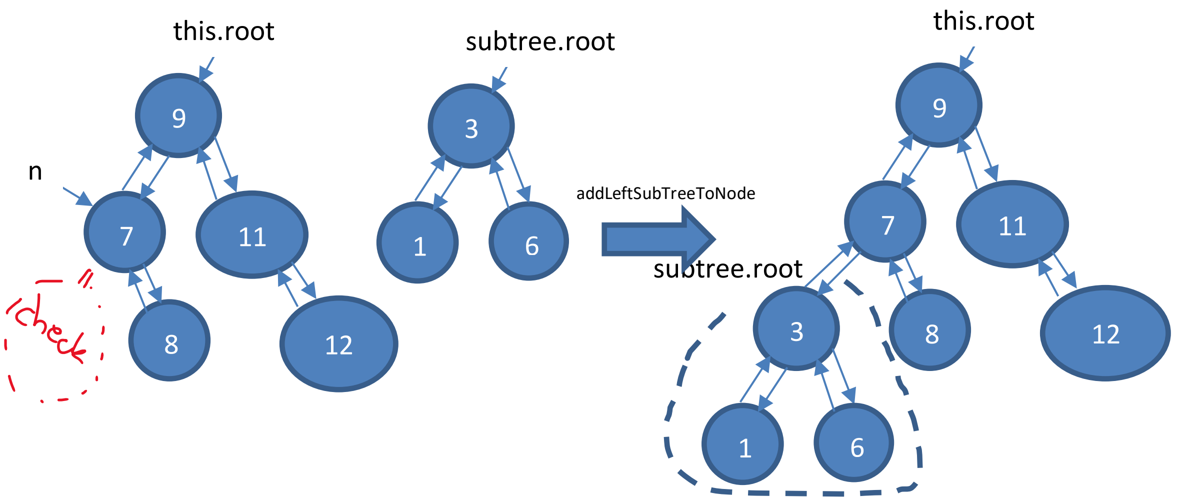
- This method tries to add an entire "subtree" as a left subtree of node n in our tree, changing our tree.
- Assume there will be no direct access to "subtree" in the future.
- Assume n originally has no left child before this method is called.
- If the "subtree" is an empty tree, this method does nothing. The method then ends.
- If our tree is empty, then our tree becomes the subtree. The method then ends.
- Check if, after the addition of "subtree", the tree will still be a binary search tree:
  - If so, link the entire subtree to node n.
  - If not, do nothing.
  - Hint:
    - Some code has been provided for this method.
    - 2 conditions needed to be checked, one is the same as the homework. You have to work out the other condition.
- Performance does not matter.
- You code must work with any node n in a binary search tree.
- Only BST.java is allowed to be modified.
  - You are NOT allowed to modify other existing methods.
  - You are allowed to create new method(s). (You won't need to).

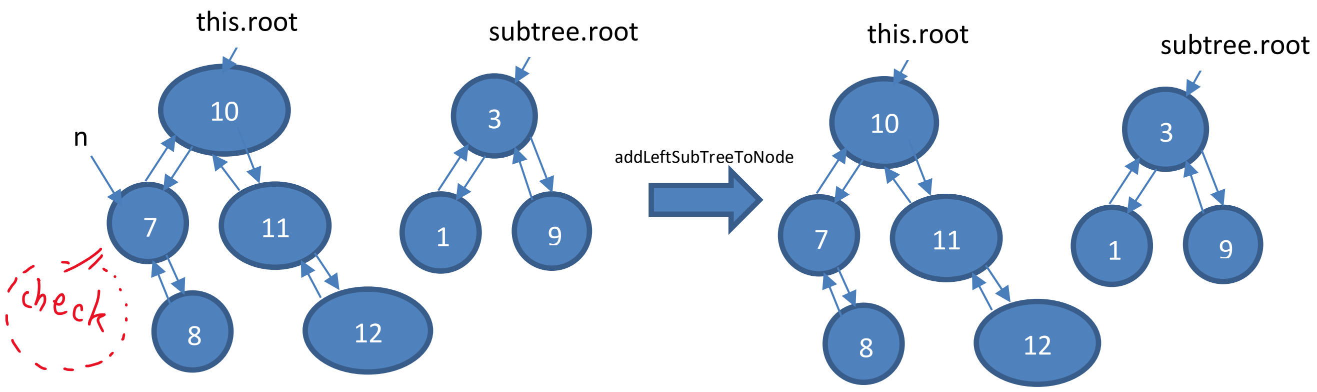
Example:



Continue next page

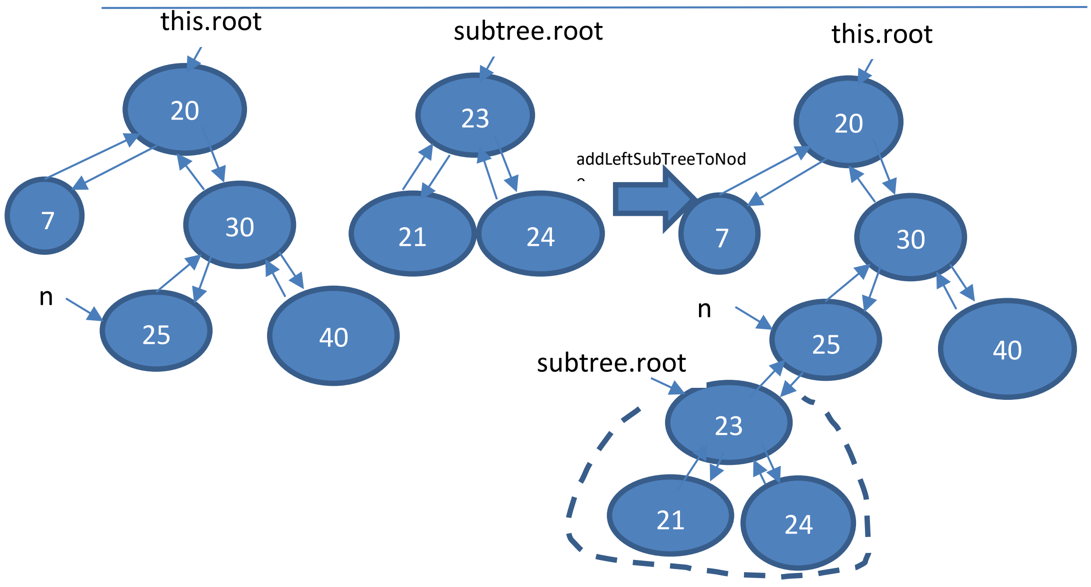
Name………………………………….…..ID……….……………………………CR58……………………



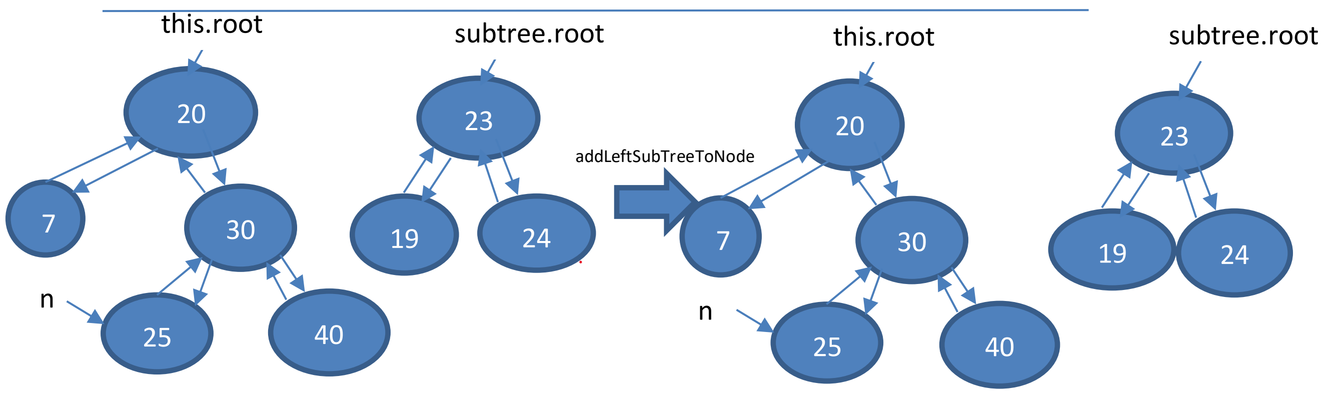


Nothing changes because we won't get a binary search tree.

.



Continue next page



Nothing changes because we won't get a binary search tree.

The JUnit tests are in BSTTest.java

| | |
|---|---|
| • testAddEmptySubTree() | 1 marks |
| • testAddToEmptyTree() | 1 marks |
| • testAddToLeftMostSuccess() | 1 marks |
| • testAddToLeftMostFail() | 1 marks |
| • testAddToMiddleSuccess() | 3 marks |
| • testAddToMiddleFail01() | 2 marks |
| • testAddToMiddleFail02() | 1 marks |