- Don't forget to set your Eclipse workspace and working set.
- You must submit the JAR file, exported (with source code), from your Eclipse project.
- You must check your JAR file to make sure all the source files (.java files) are present. It can be opened with file compression programs such as 7-zip or Winrar.
- Failure to export properly will result in your work not getting marked.

1) To submit:

   - Export your project to a JAR file, with source code.
   - Name your JAR file ID_Week05_Q2.jar. For example, 6623110021_Week05_Q2.jar
   - Submit the JAR file on MyCourseville.

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:

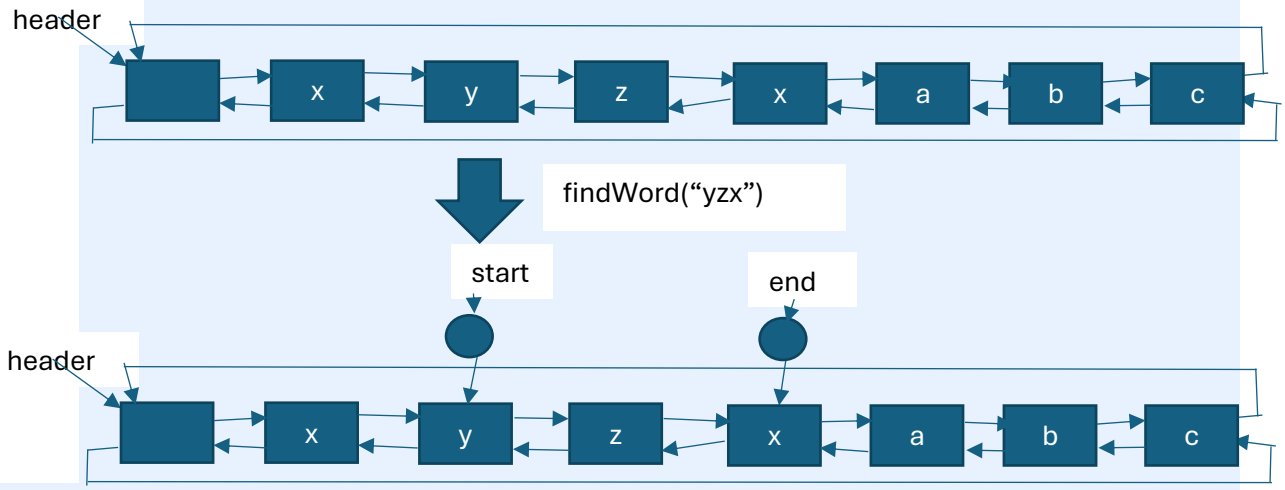
a) **(8 marks)** `public void findWord(String w) throws Exception {`

- 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.

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