3. This question involves the manipulation and analysis of a list of words. The following `WordChecker` class contains an `ArrayList<String>` to be analyzed and methods that are used to perform the analysis. You will write two methods of the `WordChecker` class.

```
public class WordChecker
{
   /** Initialized in the constructor and contains no null elements */
   private ArrayList<String> wordList;

   /**
    *  Returns true if each element of wordList (except the first) contains the previous
    *  element as a substring and returns false otherwise, as described in part (a)
    *  Precondition: wordList contains at least two elements.
    *  Postcondition: wordList is unchanged.
    */
   public boolean isWordChain()
   {  /* to be implemented in part (a) */  }

   /**
    *  Returns an ArrayList<String> based on strings from wordList that start
    *  with target, as described in part (b). Each element of the returned ArrayList has had
    *  the initial occurrence of target removed.
    *  Postconditions: wordList is unchanged.
    *      Items appear in the returned list in the same order as they appear in wordList.
    */
   public ArrayList<String> createList(String target)
   {  /* to be implemented in part (b) */  }

   // There may be instance variables, constructors, and methods that are not shown.
}
```

**GO ON TO THE NEXT PAGE.**

(a) Write the `isWordChain` method, which determines whether each element of `wordList` (except the first) contains the previous element as a substring. The following table shows two sample `isWordChain` method calls.

| `wordList` | `isWordChain` **Return Value** | **Explanation** |
|---|---|---|
| `["an", "band", "band", "abandon"]` | `true` | Each element contains the previous element as a substring. |
| `["to", "too", "stool", "tools"]` | `false` | `"tools"` does not contain the substring `"stool"`. |

Complete the `isWordChain` method.

```
/**
 * Returns true if each element of wordList (except the first) contains the previous
 * element as a substring and returns false otherwise, as described in part (a)
 * Precondition: wordList contains at least two elements.
 * Postcondition: wordList is unchanged.
 */
public boolean isWordChain()
```

**GO ON TO THE NEXT PAGE.**

(b) Write the `createList` method, which creates and returns an `ArrayList<String>`. The method identifies strings in `wordList` that start with `target` and returns a new `ArrayList` containing each identified string without the starting occurrence of `target`. Elements must appear in the returned list in the same order as they appear in `wordList`.

Consider an example where `wordList` contains the following strings.

```
["catch", "bobcat", "catchacat", "cat", "at"]
```

The following table shows the `ArrayList` returned by some calls to `createList`. In all cases, `wordList` is unchanged.

| **Method Call** | `ArrayList` **Returned by** `createList` | **Explanation** |
|---|---|---|
| `createList("cat")` | `["ch", "chacat", ""]` | Only `"catch"`, `"catchacat"`, and `"cat"` begin with `"cat"`. |
| `createList("catch")` | `["", "acat"]` | Only `"catch"` and `"catchacat"` begin with `"catch"`. |
| `createList("dog")` | `[]` | None of the words in `wordList` begin with `"dog"`. |

Complete the `createList` method.

```
/**
 * Returns an ArrayList<String> based on strings from wordList that start
 * with target, as described in part (b). Each element of the returned ArrayList has had
 * the initial occurrence of target removed.
 * Postconditions: wordList is unchanged.
 *       Items appear in the returned list in the same order as they appear in wordList.
 */
public ArrayList<String> createList(String target)
```