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.

© 2024 College Board.
Visit College Board on the web: collegeboard.org.

(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()
```

Begin your response at the top of a new page in the separate Free Response booklet and fill in the appropriate circle at the top of each page to indicate the question number. If there are multiple parts to this question, write the part letter with your response.

GO ON TO THE NEXT PAGE.

© 2024 College Board.
Visit College Board on the web: collegeboard.org.

(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)
```

Begin your response at the top of a new page in the separate Free Response booklet and fill in the appropriate circle at the top of each page to indicate the question number. If there are multiple parts to this question, write the part letter with your response.

Class information for this question

```
public class WordChecker

private ArrayList<String> wordList

public boolean isWordChain()
public ArrayList<String> createList(String target)
```

GO ON TO THE NEXT PAGE.

4. This question involves a path through a two-dimensional (2D) array of integers, where the path is based on the values of elements in the array. When an element of the 2D array is accessed, the first index is used to specify the row and the second index is used to specify the column. The following `Location` class represents a row and column position in the 2D array.

```
public class Location
{
   private int theRow;
   private int theCol;

   public Location(int r, int c)
   {
      theRow = r;
      theCol = c;
   }

   public int getRow()
   {  return theRow;  }

   public int getCol()
   {  return theCol;  }
}
```

GO ON TO THE NEXT PAGE.

© 2024 College Board.
Visit College Board on the web: collegeboard.org.

Question 3: Array / ArrayList — 9 points

Canonical solution

(a) **3 points**

```
public boolean isWordChain()
{
   for (int i = 1; i < wordList.size(); i++)
   {
      String current = wordList.get(i);
      String previous = wordList.get(i - 1);

      if (current.indexOf(previous) == -1)
      {
         return false;
      }
   }
   return true;
}
```

(b) **6 points**

```
public ArrayList<String> createList(String target)
{
   ArrayList<String> result = new ArrayList<String>();

   for (String current : wordList)
   {
      if (current.indexOf(target) == 0)
      {
         String newStr = current.substring(target.length());
         result.add(newStr);
      }
   }

   return result;
}
```

(b) `createList`

	Scoring Criteria	Decision Rules	
4	Declares and constructs an `ArrayList<String>`	Responses **will not** earn the point if they • fail to declare an `ArrayList`	**1 point**
5	Accesses all elements of `wordList` (*no bounds errors*)	Responses **can** still earn the point even if they • return early, as long as bounds and indices would otherwise support accessing all elements Responses **will not** earn the point if they • fail to access elements of `wordList` correctly	**1 point**
6	Identifies strings that begin with `target` (*in the context of an* `if`)	Responses **can** still earn the point even if they • access elements of `wordList` incorrectly Responses **will not** earn the point if they • call `String` methods incorrectly • identify all strings that contain `target` • use the `substring` method without a guard against an element too short to contain `target`	**1 point**
7	Constructs a `String` that is a copy of an element of the list with the correct number of initial characters removed	Responses **can** still earn the point even if they • make a copy of a `wordList` element that does not start with `target` or is not long enough to contain `target`	**1 point**
8	Adds to the constructed list at least one `String` based on an element of the original list	Responses **can** still earn the point even if they • add an incorrectly constructed `String` • have not constructed a list Responses **will not** earn the point if they • call `add` incorrectly	**1 point**

© 2024 College Board

9	Returns list containing all and only identified and revised strings in the appropriate order (*algorithm*)	Responses **can** still earn the point even if they • incorrectly identify strings beginning with `target` • call `add` incorrectly Responses **will not** earn the point if they • add the original, unrevised element to the list to be returned • modify `wordList` or any of its elements • return an incorrect value due to an early return	**1 point**
		Total for part (b)	**6 points**
		Total for question 3	**9 points**

Note that a correct part (b) solution could replace the `indexOf` call in the `if` statement with:

```
if (current.length() >= target.length() &&
    current.substring(0, target.length()).equals(target))
```

Applying the Scoring Criteria

Apply the question scoring criteria first, which always takes precedence. Penalty points can only be deducted in a part of the question that has earned credit via the question rubric. No part of a question (a, b, c) may have a negative point total. A given penalty can be assessed only once for a question, even if it occurs multiple times or in multiple parts of that question. A maximum of 3 penalty points may be assessed per question.

1-Point Penalty

v) Array/collection access confusion (`[] get`)
w) Extraneous code that causes side-effect (e.g., printing to output, incorrect precondition check)
x) Local variables used but none declared
y) Destruction of persistent data (e.g., changing value referenced by parameter)
z) Void method or constructor that returns a value

No Penalty

- Extraneous code with no side-effect (e.g., valid precondition check, no-op)
- Spelling/case discrepancies where there is no ambiguity*
- Local variable not declared provided other variables are declared in some part
- `private` or `public` qualifier on a local variable
- Missing `public` qualifier on class or constructor header
- Keyword used as an identifier
- Common mathematical symbols used for operators (× • ÷ ≤ ≥ <> ≠)
- `[]` vs. `()` vs. `<>`
- `=` instead of `==` and vice versa
- `length/size` confusion for array, `String`, `List`, or `ArrayList`; with or without `( )`
- Extraneous `[]` when referencing entire array
- `[i,j]` instead of `[i][j]`
- Extraneous size in array declaration, e.g., `int[size] nums = new int[size];`
- Missing `;` where structure clearly conveys intent
- Missing `{ }` where indentation clearly conveys intent
- Missing `( )` on parameter-less method or constructor invocations
- Missing `( )` around `if` or `while` conditions

Spelling and case discrepancies for identifiers fall under the "No Penalty" category only if the correction can be* ***unambiguously* *inferred from context, for example, "ArayList" instead of "ArrayList". As a counterexample, note that if the code declares* **`"int G=99, g=0;"`**, *then uses* **`"while (G < 10)"`** *instead of* **`"while (g < 10)"`**, *the context does* ***not*** *allow for the reader to assume the use of the lower-case variable.*

© 2024 College Board