COMP1006/1406 – Fall 2023

Submit a single file, called `SListArray.java` to Gradescope.
This assignment has 10 marks.

A String List [10 marks]

In this problem you will implement a variant of the List ADT. In particular you will implement the String-List ADT, in a concrete class called `SListArray`, based on the provided abstract `Slist` class.

```
public abstract class SList{
  public abstract String get(int position);
  public abstract String set(int position, String element);
  public abstract String add(int position, String element);
  public abstract String remove(int position);
  public abstract int size();

  public abstract void append(SList anotherSList);

  public abstract SList   commonStrings();
  public abstract SList[] commonStrings(int n);
}
```

The specification of the methods are provided in the javadoc comments in `SList.java` (and also the provided `SList.html` file).

Your concrete child class will be called `SListArray`. As the name suggests, you will need to use an **array** to store the data for your list. You are **NOT** allowed to use an `ArrayList` or any other class that Java provides to store your list. Your `SListArray` class must have the following constructors

```
public SListArray()
 // creates an empty list

public SListArray(String[] elements)
 // creates a list with the strings in elements
 // the ordering and size of this list will be the same as in elements
```

For the `commonStrings()` methods, how do you find the most commonly appearing strings in the list? A standard way of doing this, **and the way you must do this**, is to create a dictionary of word-frequency pairs. That is, for any string (word), you keep track of how many times it appears (frequency) in the list.

First, you iterate over the list and for each string you first check if it already appears in the dictionary or not. If it is already there, you change its value (frequency) by incrementing by 1. If it is not there, you add the word with frequency 1 to the dictionary.

Once the dictionary is created, you then iterate over all values (frequencies) to find the largest frequency. After you find the largest frequency, you iterate over the dictionary again to build a list with all the words with that given frequency. There may be more than one word that appears the maximal number of times.

There is slightly more work involved when you need to find words with the n top frequencies (`commonStrings(int n)`). In this case, after you create the dictionary, instead of then looking for

the largest frequency, you can build a `Set` of all the frequencies in the dictionary. If you choose a good `Set` data structure (concrete class) you can then read off the first n values in sorted order. Hint: https://www.mastercoding.org/java-programs/java-treeset-reverse-order/

Here is a recap of some of the restrictions for this assignment

- You **must** use a partially-filled array to store your list. That is, use an array that is larger than the list. If you ever run out of room make a bigger array. This is called the backing array.
  - Start with an initial capacity of 8.
  - If you need more capacity when adding, double the capacity. That is, make a new backing array that is twice the size of the current one.
  - If you need more capacity when appending, compute how much capacity you need (this.size() + anotherList.size()) and double that.
  - We will not worry about *shrinking* the backing array. (If you are eager to do this though, each time you remove a string, if the current size is less than 1/4 of the capacity, then reduce the capacity by 1/2. That is, make a new backing array that is half the size as the current array.)
- You **must** use a `HashMap` (dictionary) for your `commonStrings()` methods
- You **should** use an appropriate `Set` to collect your frequencies for `commonStrings(int)`.
- You are **NOT** allowed to use an `ArrayList` in this assignment. (The assignment is, essentially, asking you implement this!)

Submit your `SListArray.java` file to gradescope.