COMPUTER SCIENCE A

SECTION II

Time—1 hour and 45 minutes

Number of questions—4

Percent of total grade—50

Directions: SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

Notes:

- Assume that the classes listed in the Quick Reference found in the Appendix have been imported where appropriate.
- Unless otherwise noted in the question, assume that parameters in method calls are not `null` and that methods are called only when their preconditions are satisfied.
- In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods may not receive full credit.

Consider a method of encoding and decoding words that is based on a *master string*. This master string will contain all the letters of the alphabet, some possibly more than once. An example of a master string is `"sixtyzipperswerequicklypickedfromthewovenjutebag"`. This string and its indexes are shown below.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
s	i	x	t	y	z	i	p	p	e	r	s	w	e	r	e	q	u	i	c	k	l	y	p

24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
i	c	k	e	d	f	r	o	m	t	h	e	w	o	v	e	n	j	u	t	e	b	a	g

An encoded string is defined by a list of *string parts*. A string part is defined by its starting index in the master string and its length. For example, the string `"overeager"` is encoded as the list of string parts [(37, 3), (14, 2), (46, 2), (9, 2)] denoting the substrings `"ove"`, `"re"`, `"ag"`, and `"er"`.

String parts will be represented by the `StringPart` class shown below.

```
public class StringPart
{
  /** @param start  the starting position of the substring in a master string
   *  @param length  the length of the substring in a master string
   */
  public StringPart(int start, int length)
  {  /* implementation not shown */  }


  /** @return  the starting position of the substring in a master string
   */
  public int getStart()
  {  /* implementation not shown */  }


  /** @return  the length of the substring in a master string
   */
  public int getLength()
  {  /* implementation not shown */  }

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

The class `StringCoder` provides methods to encode and decode words using a given master string. When encoding, there may be multiple matching string parts of the master string. The helper method `findPart` is provided to choose a string part within the master string that matches the beginning of a given string.

```
public class StringCoder
{
  private String masterString;

  /** @param master the master string for the StringCoder
   *         Precondition: the master string contains all the letters of the alphabet
   */
  public StringCoder(String master)
  {  masterString = master;  }


  /** @param parts an ArrayList of string parts that are valid in the master string
   *         Precondition: parts.size() > 0
   *  @return the string obtained by concatenating the parts of the master string
   */
  public String decodeString(ArrayList<StringPart> parts)
  {  /* to be implemented in part (a) */  }


  /** @param str the string to encode using the master string
   *         Precondition: all of the characters in str appear in the master string;
   *                       str.length() > 0
   *  @return a string part in the master string that matches the beginning of str.
   *          The returned string part has length at least 1.
   */
  private StringPart findPart(String str)
  {  /* implementation not shown */  }


  /** @param word the string to be encoded
   *         Precondition: all of the characters in word appear in the master string;
   *                       word.length() > 0
   *  @return an ArrayList of string parts of the master string that can be combined
   *          to create word
   */
  public ArrayList<StringPart> encodeString(String word)
  {  /* to be implemented in part (b) */  }

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

(a) Write the `StringCoder` method `decodeString`. This method retrieves the substrings in the master string represented by each of the `StringPart` objects in `parts`, concatenates them in the order in which they appear in `parts`, and returns the result.

Complete method `decodeString` below.

```
/** @param parts an ArrayList of string parts that are valid in the master string
 *          Precondition: parts.size() > 0
 *  @return the string obtained by concatenating the parts of the master string
 */
public String decodeString(ArrayList<StringPart> parts)
```

(b) Write the `StringCoder` method `encodeString`. A string is encoded by determining the substrings in the master string that can be combined to generate the given string. The encoding starts with a string part that matches the beginning of the word, followed by a string part that matches the beginning of the rest of the word, and so on. The string parts are returned in an array list in the order in which they appear in `word`.

The helper method `findPart` must be used to choose matching string parts in the master string.

Complete method `encodeString` below.

```
/** @param word the string to be encoded
 *          Precondition: all of the characters in word appear in the master string;
 *                        word.length() > 0
 *  @return an ArrayList of string parts of the master string that can be combined
 *          to create word
 */
public ArrayList<StringPart> encodeString(String word)
```