COMPUTER SCIENCE
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.

Write your answers in pencil only in the booklet provided.

Notes:

- Assume that the classes in the Quick Reference have been imported where needed.
- Unless otherwise stated, 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.

1. Consider a system for processing names and addresses from a mailing list. A `Recipients` class will be used as part of this system. The lines in the mailing list are stored in an `Array String`, a private instance variable in the `Recipients` class. The blank line that separates recipients in the mailing list is stored as the empty string in this list, and the final element in the list is an empty string.
A portion of the mailing list is shown below, with the corresponding part of the `Array`.

```
Mr. J. Adams
6 Rose St.
Ithaca, NY 14850

Jack S. Smith
12 Posy Way
Suite 201
Glendale, CA 91203

Ms. M.K. Delgado
2 River Dr.
New York, NY 10013

...
```

0	1	2	3	4
"Mr. J. Adams"	"6 Rose St."	"Ithaca, NY 14850"	""	"Jack S. Smith"

5	6	7	8	9
"12 Posy Way"	"Suite #201"	"Glendale, CA 91023"	""	"Ms. M.K. Delgado"

10	11	12	
"2 River Dr."	"New York, NY 10013"	""	...

The Recipients class that processes this data is shown below.

```
public class Recipients
{
    /** The list of lines in the mailing list */
    private      String[] lines;

    /** Constructor. Fill lines with mailing list data.
     *  Postcondition:
     *    - Each element in lines is one line of the mailing list.
     *    - Lines appear in the list in the same order
     *      that they appear in the mailing list.
     *    - Blank line separators in the mailing list are stored
     *      as empty strings.
     */
    public Recipients()
    { /* implementation not shown */ }

    /** Postcondition: Returns the city contained in the cityZip
     *                 string of an address.
     *  @param  cityZip contains the city, state, and zipcode
     *  line of an address
     *  @return the city substring contained in cityZip
     */
    public String extractCity(String cityZip)
    { /* to be implemented in part (a) */ }

    /** Precondition: The recipient name is the first line of each
     *                label on the mailing list.
     *  Postcondition: Prints a list of recipient names to console,
     *                 one per line.
     */
    public void printNames()
    { /* to be implemented in part (b) */ }
```

```
    /** Postcondition: Returns the address of the recipient with
     *                 the specified name.
     *  @param name a name in the lines ArrayList
     *  @return the address of the recipient with the given name
     */
    public String getAddress(String name)
    {/* to be implemented in part (c) */}

    //Other methods are not shown.
}
```

(a) Write the `extractCity` method of the `Recipients` class. In the `cityZip` parameter the city is followed by a comma, then one blank space, then two capital letters for a state abbreviation, then a space and 5-digit zip code. For example, if `cityZip` is `"Ithaca, NY 14850"`, the method call `extractCity(cityZip)` should return `"Ithaca"`.

Information repeated from the beginning of the question

```
public class Recipients

private     String[] lines
public Recipients()
public String extractCity(String cityZip)
public void printNames()
public String getAddress(String name)
```

Complete method `extractCity` below.

```
/** Postcondition: Returns the city contained in the cityZip
 *                 string of an address.
 *  @param  cityZip contains the city, state, and zipcode
 *  line of an address
 *  @return the city substring contained in cityZip
 */
public String extractCity(String cityZip)
```

(b) Write the `printNames` method of the `Recipients` class. Method `printNames` prints the names of all recipients to the console, one per line. For the sample part of the mailing list shown at the beginning of the question, the output for `printNames` would be:

```
Mr. J. Adams
Jack S. Smith
Ms. M.K. Delgado
```

Complete method printNames below.

```
/** Precondition: The recipient name is the first line of each
 *                label on the mailing list.
 *  Postcondition: Prints a list of recipient names to console,
 *                one per line.
 */
public void printNames()
```

(c) Write the getAddress method of the Recipients class. This method should return a string that contains only the address of the corresponding name parameter. For example, if name is "Jack S. Smith", a string containing the three subsequent lines of his address should be returned. This string should contain line breaks in appropriate places, including after the last line of the address. This ensures that the address will have the proper address format when printed by a client class. return blank string if not found.

Complete method getAddress below.

```
/** Postcondition: Returns the address of the recipient with
 *                the specified name.
 *  @param name a name in the lines ArrayList
 *  @return the address of the recipient with the given name
 */
public String getAddress(String name)
```

```
public String extractCity( String cityZip) {
        int  i = cityZip.indexOf( "," );
        String s =  cityZip.substring (0, i);
        return s;
}
```

```
public void printNames() {

System.out.println(lines[0]);
for (int i=1; i< lines.length-1; i++) {
      if ( lines[i].equals(""))
            System.out.println (lines[i+1]);
 } // end for
 } // end method
```

```
public String getAddress ( String name) {

String s= "";
for ( int i=0; i< lines.length ; i++) {
      if (lines[i].equals (name) ) {
            i++;
            while ( ! lines[i].equals("") )
                  s+= lines[i] + "/n";
            return s;    // or break;
      } //end if
 } // end for
 return s;
} // end method
```