

Question 1. [20 MARKS]

1. Write a class called `Anniversary` that customizes the `Date` class, and contains a `String` instance variable called `spouseName`.

```
public class Anniversary extends Date {
    private String spouseName;
}
```

2. Given an integer called `hoursLeft`, write a `while` loop that decrements `hoursLeft` by 1 each iteration and continues looping as long as `hoursLeft` is not negative.

```
while (hoursLeft >=0) {
    hoursLeft--;
}
```

3. Given a `StringTokenizer` object called `dispenser`, write a `while` loop that prints all of `dispenser`'s tokens to the screen, one per line.

```
while (dispenser.hasMoreTokens()) {
    System.out.println(dispenser.nextToken());
}
```

4. Given an array called `bookTitles` that contains at least 1 `Object`, write an expression that evaluates to `true` if the first element of `bookTitles` is a `String` object.

```
bookTitles[0] instanceof String
```

5. Write a single statement that declares and instantiates an array of four `String` objects called `seasons`, and initializes it to the values "Spring", "Summer", "Autumn" and "Winter".

```
String[] seasons = {"Spring", "Summer", "Autumn", "Winter"};
```

6. Given an array `studentNumbers`, write an expression that refers to the last element of the array.

```
studentNumbers[studentNumbers.length - 1]
```

7. Write a header for a `static` method called `flatten` that has 2 parameters: a two-dimensional array of `Strings` called `squareArray`, and a one-dimensional array of `Strings` called `flatArray`. `flatten` does not return anything. Leave the body of this method empty (the implementation is asked for in the next question).

```
public static void flatten(String[][] squareArray, String[] flatArray)
```

8. Assume that `squareArray` refers to a 4x4 array of `Strings` and that `flatArray` refers to a `String` array of length 16. Write code that copies the elements from `squareArray` into `flatArray`.

```
int counter = 0;
for (int i = 0; i < 4; i++) {
    for (int j = 0; j < 4; j++) {
        flatArray[counter] = squareArray[i][j];
        counter++;
    }
}
```

9. Given a rectangular 2D array called `matrix`, write an expression that evaluates to `true` if `matrix` is a square array. (You may assume that there is at least one row.)

```
matrix.length == matrix[0].length
```

10. In some class, there is an instance variable called `students` that refers to a one-dimensional array of `Student` objects. Write a method called `doubleCapacity` that will be called when `students` becomes full. After calling `doubleCapacity`, `students` should refer to an array that is twice as big as the original array, and should contain all of the elements of the original array in their original positions.

```
public void doubleCapacity() {
    Student[] temp = new Student[2 * students.length];
    for (int i = 0; i < students.length; i++) {
        temp[i] = students[i];
    }
    students = temp;
}
```

Question 2. [20 MARKS]

On the opposite page, some of the code fragments have errors. For each fragment:

- If the fragment has a compile error, then place a checkmark ✓ in the first column.
- If the fragment compiles, but has a run-time error, then place a checkmark ✓ in the second column.
- If there is no error, then in the third column briefly describe the result of executing the fragment. If there is an error then *do not write in the last column*. If you know there is no error, but you are unsure of the result, then place a checkmark ✓ in the last column.

You should assume the following:

- All necessary `import` statements have been done.
- All classes are in separate files.
- All code fragments appear in separate classes and methods.

Code Fragment	Compile error?	Run-time error?	Result, if successful?
<code>char a = 3 + 'c';</code>			a's value: 'f'
<code>boolean b = 1 && 0;</code>	✓		b's value:
<code>String c = "final exam"; char d = c.charAt(c.length - 1);</code>	✓		d's value:
<code>String e = "testing 123"; e = e.substring(1, 4); char f = e.charAt(3);</code>		✓	f's value:
<code>Integer g = new Integer(25); System.out.println(g * 3);</code>	✓		output of System.out.println:
<code>double h = int(3 * 5 / 2);</code>	✓		h's value
<code>JFrame j = new JFrame("Final"); JFrame k = new JFrame("Exam"); j = k; k = j; String t = k.getTitle();</code>			t's value: "Exam"
<code>Date[] d = new Date[5]; int month = d[0].getMonth();</code>		✓	month's value:
<code>int i = 0; int[] a = {1,2,3,4,5}; while (i <= a.length) { a[i] = a[i] + 1; }</code>		✓	a's contents:
<code>int[][] z = {{1,2},{4,5},{5,6}}; for(int i = 0; i < z.length; i++) { z[i][i] = z[i][i] * 2; }</code>		✓	z's contents:

Question 3. [20 MARKS]**Part (a)** [6 MARKS] Complete the following code.

```
/** A song with a title and an integer length (in seconds). */
public class Song {

    // Declare any necessary instance variables here.

    private String title;
    private int length;

    /** A new song with title t and length len. */
    public Song(String t, int len) {

        title = t;
        length = len;

    }

    /** Return the title of this song. */
    public String getTitle() {

        return this.title;

    }

    /** Set the title of the song to t. */
    public void setTitle(String t) {

        this.title = t;

    }

    /** Get the length of the song. */
    public int getLength() {

        return length;

    }
}
```

Part (b) [14 MARKS] Complete the following code. You will be marked on both design and correctness.

```
/**
 * An album with a list of songs. The first song is at index 0. An album has
 * both a maximum number of songs and a maximum length (in seconds).
 */
public class Album {

    // Declare any necessary instance variables here.

    private Song[] songs;
    private int lastSong; // index of the last song, or -1 if there are none
    private int maxLength;
    private int currentLength;
    private int numSongs;

    /** A new album that can hold up to n songs and s seconds worth of songs. */
    public Album(int n, int s) {

        songs = new Song[n];
        maxLen = s;
        numSongs = 0;

    }

    /** Return the index of the song with title t, or -1 if there is no song with that title. */
    public int getSongIndex(String t) {

        int i = 0;
        while (i != numSongs && !songs[i].getTitle().equals(t)) {
            i++;
        }
        if (i == numSongs) {
            return -1;
        } else {
            return i;
        }
    }
}
```

```
/** Return true if there are seconds left on this album and space for another song. */
public boolean isRoom() {

    return currentLength < maxLength && lastSong < songs.length;

}

/**
 * Add s to the end of the list of songs, unless it will not fit or there is already a
 * song with that title on the album. Return whether s was successfully added.
 */
public boolean addSong(Song s) {

    int index = getSngIndex(s.getTitle());
    if (index == -1 && isRoom() && currentLength + s.getLength() <= maxLength) {
        lastSong++;
        songs[numSongs] = s;
        currentLength = currentLen + s.getLength();
        return true;
    }
    else {
        return false;
    }

}

/**
 * Remove the song with title t from the album (if it is currently on the album) and
 * return whether the song was removed. The order of the remaining songs does not change.
 */
public boolean removeSong(String t) {

    int index = getSngIndex(t);
    if (index == -1) {
        return false;
    } else {
        currentLength = currentLength - songs[index].getLength();
        for (int i = index; i != numSongs - 1; i++) {
            songs[i] = songs[i + 1];
        }

        songs[numSongs - 1] = null;
        numSongs--;
        return true;
    }

}
}
```

Question 4. [10 MARKS]

Part (a) [2 MARKS] An array of `String` arrays is declared and initialized as follows:

```
String[][] firstNames = new String[3][];
firstNames[0] = new String[1];
firstNames[1] = new String[3];
firstNames[2] = new String[5];
```

How many names will fit into the 2D array referred to by `firstNames`? **9**

MARKING: Either right or wrong: 0 or 2 marks

Part (b) [8 MARKS] Consider class A:

```
public class A {
    public static char[][] smart(char[][] inputArray, int r, int c) {
        char[][] t = new char[c][r];
        for (int i = 0; i < r; i++) {
            for (int j = 0; j < c; j++) {
                t[j][i] = inputArray[i][j];
            }
        }

        return t;
    }

    public static void printCharArray(char[][] charArray) {
        for (int i= 0; i < charArray.length; i++){
            for (int j=0; j < charArray[i].length; j++) {
                System.out.print(charArray[i][j] + " ");
            }

            System.out.println();
        }
    }
}
```

What is the output of the following set of statements?

```
char[][] kp = {{'1','4','7','*'}, {'2','5','8','0'}, {'3','6','9','#'}};
char[][] tkp = A.smart(kp, 3, 4);
A.printCharArray(tkp);
```

```
1 2 3
4 5 6
7 8 9
* 0 #
```

Question 5. [20 MARKS]**Part (a)** [10 MARKS]

Write a `public static void` method called `diagonal` that has a square two-dimensional array of integers as its parameter and sets all the values along the two diagonals to `-1`. (The diagonals go from the upper-left corner down to the lower-right corner, and the upper-right corner down to the lower-left corner.) You may assume the parameter is not null and refers to a square array. Your code should be as simple as possible.

Also write a Javadoc comment for your method, including an `@param` tag.

```
public class Diagonal {

    /**
     * Set the values along the diagonals of square array in to -1.
     * @param in the square 2D array
     */
    public static void diagonal(int[][] in) {

        // Pretty solution:
        for (int i = 0; i < in.length; i++) {
            in[i][i] = -1;
            in[i][in.length - i - 1] = -1;
        }

        // Not as simple a solution:
        for (int i = 0; i < in.length; i++) {
            for (int j = 0; j < in.length; j++) {
                if (i == j) {
                    in[j][i] = -1;
                    in[j][in.length - i - 1] = -1;
                }
            }
        }
    }
}
```

Part (b) [10 MARKS]

Write a good set of JUnit test methods to test `diagonal`. You probably will need a `private` helper method to compare arrays. *Only test with square arrays.*

```
import junit.framework.TestCase;

/** Test method diagonal. */
public class TestDiagonal extends TestCase {
```

Question 6. [20 MARKS]

Part (a) [10 MARKS]

Consider the following array of **Strings**.

```

+=====+=====+=====+=====+=====+=====+=====+
| Sleepy | Happy  | Grumpy | Dopey  | Sneezy | Bashful | Doc   |
+=====+=====+=====+=====+=====+=====+=====+

```

Perform the first four iterations of insertion and selection sort, and draw a snapshot of the array contents in each case in the array diagrams shown below. In this case, four iterations means that four elements have been considered and sorted in alphabetical order, with the remaining three left to go. (You may assume that **String**'s **compareTo** method has been used to determine ordering.)

Insertion Sort after 4 iterations:

```

+=====+=====+=====+=====+=====+=====+=====+
| Dopey  | Grumpy | Happy  | Sleepy | Sneezy | Bashful | Doc   |
+=====+=====+=====+=====+=====+=====+=====+

```

Selection Sort after 4 iterations:

```

+=====+=====+=====+=====+=====+=====+=====+
| Bashful | Doc   | Dopey  | Grumpy | Sneezy | Sleepy | Happy |
+=====+=====+=====+=====+=====+=====+=====+

```

Part (b) [10 MARKS]

Write a `public static` method called `alphaOrder` that takes in a `String` parameter called `inputString`. This method returns a `String` that has the same length and contains the same letters as `inputString`, but with the letters in alphabetical order.

Characters in `inputString` may occur more than once. You may assume that all the characters of `inputString` are lowercase letters. You will be marked on both design and correctness.

```
public static String alphaOrder(String input) {  
  
    String returnString = "";  
    while (input.length() != 0) {  
  
        int smallestLetter = 0;  
        for (int i = 0; i < input.length(); i++) {  
            if (input.charAt(i) < input.charAt(smallestLetter)) {  
                smallestLetter = i;  
            }  
        }  
  
        returnString += input.charAt(smallestLetter);  
        input = input.substring(0, smallestLetter) + input.substring(smallestLetter + 1);  
  
    }  
  
    return returnString;  
}
```

Question 7. [20 MARKS]

This question is about input and output. Assume that all appropriate classes have been imported.

Part (a) [8 MARKS] Complete this method:

```
/** Return the number of lines in the file named fileName. */
public static int numLines(String fileName) throws IOException {

    int count = 0;
    BufferedReader br = new BufferedReader(new FileReader(fileName));
    String line = br.readLine();

    while (line != null) {
        count = count + 1;
        line = br.readLine();
    }

    return count;
}
```

Part (b) [12 MARKS]

Complete the method below so that it prints the prompt "Enter an integer: ", waits for a user to type an integer, triples the entered value, and writes the result to a file named fileName.

```
public static void writeInt(String fileName) throws IOException {

    int input;
    PrintStream fw = new PrintStream(new FileOutputStream(fileName));
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    System.out.print("Enter an integer: ");
    input = Integer.parseInt(br.readLine());
    fw.println(3 * input);
}
```