Reichman University

Efi Arazi School of Computer Science

Introduction to Computer Science

Midterm Examination Sample

- The exam lasts 2 hours. There will be no time extension.
- Use your time efficiently. If you get stuck somewhere, leave the question and move on to another question.
- Use of digital devices, books, lecture notes, and anything other than this exam form is forbidden. All the materials that you need for answering this exam are supplied with the exam.
- Answer all questions on the current exam form.
- Answer all the questions on the exam pages. **Don't write anything on the back of the pages**. Only the front pages are scanned for grading. You will get some blank pages for draft (טיוטה).
- You can answer any question in either English or Hebrew.
- If you feel a need to make an assumption, you may do so as long as the assumption is reasonable and clearly stated.
- If you can't give a complete answer, give a partial answer. A partial answer will award partial points.
- If you are asked to write code and you feel that you can't write it, you may describe what you wish to do in natural language (English or Hebrew). A good explanation will award partial credit.
- If you are asked to write code that operates on some input, there is no need to validate the input unless you are explicitly asked to do so. Likewise, if you are asked to write a method that operates on some arguments, there is no need to validate the arguments unless you are explicitly asked to do so.
- There is no need to document the code that you write, unless you want to communicate something to us.
- The code that you will write in the exam will be judged, among other things, on its conciseness, elegance, and efficiency. Unnecessarily long or cumbersome code will cause loss of points, even if it provides the correct answer.
- No points will be taken for trivial syntax errors. For example, instead of writing `System.out.println(x)` you can write `println(x)`.

Good Luck!

IMPORTANT: You should view this exam sample as yet another homework in this course. It's a worked out homework, for which the solution is supplied. Solving the exam questions on your own, without looking at the supplied solutions, is the best way to practice for the midterm exam.

1. Consider the following program:

```
// This method receives a square matrix (in which the number of rows and columns is the same)
// and computes and returns something.
public static boolean mystery(int[][] m) {
    int n = m.length-1;
    int a1 = 0, a2 = 0;
    for (int i = 0; i <= n; i++) {
        a1 = a1 + m[i][i];
        a2 = a2 + m[i][n-i];
    }
    return a1 == a2;
}
```

a. (10 points) What does this method compute? What value does it return? We expect you to give a general answer, such as: "The method computes the number of columns in the matrix which do not contain a zero". Of course, this method does something else. What does it do? Write a short answer of no more than two sentences.

b. (5 points) The method above assumes, but does not check, that the given array has an equal number of rows and columns. What would happen if we run this code with an array for which the number of rows is different from the number of columns? Write a short and precise answer of no more than two sentences.

All the remaining questions in the exam deal with the class `Sets`, documented in two help pages.

The `Sets` class is a library providing various services for performing operations on sets containing integer numbers. A set is defined as a collection of elements without regard to order and without repetition. For example, the collection 3, 6, 3, 2, 1, 2 is not a set, since at least one of the values is repeated. In contrast, {3, 6, 2, 1} is a set. The sets {3, 6, 2, 1} and {1, 2, 3, 6} are considered equal, since the order of the elements in a set is meaningless.

Before answering or even reading the remaining questions, we recommend that you spend about ten minutes reading the two help pages titled "The `Sets` Class".

Answer question 2 or question 3 (answer 1 question only)

2. (13 points) The method `print` receives an array that represents a set, and prints it. Write the method.

```
/** Prints the elements of the given set, starting and ending with curly brackets.
 *  For example, if the array contains the elements 7,2,1,5, prints { 7, 2, 1, 5 }  */
public static void print(int[] set) {
   // Write your code here:
```

3. (13 points) The method `elementOf` checks if a given value appears in a given array. For example, if the given array is [7, 2, 5, 1] and the given value is 2, the method returns `true`. If the value is 3, the method returns `false`. Write the method.

```
/** Checks if the given element exists in the given set. */
public static boolean elementOf(int e, int[] set) {
    // Write your code here:
```

4. (10 points) The method `random` returns a value chosen at random from the given set. For example, if the set is {3, 5, 1}, the method could return 5 on one run, 3 on another run, 5 again on another run, and so on. Write the method.

```
/** Returns a random element chosen from the given set. */
public static int random (int[] set) {
    // Write your code here:
```

Note: Assume that all the functions of the `Sets` class have been implemented correctly.
You can call these functions in your solutions, if you think it makes sense.

5. (25 points) The method `buildSet` accepts an array and returns another array containing all the elements in the given array, without repetitions. Write the method.

```
/** Builds a set from a given array.
 *  For example, if the given array is [7, 2, 7, 1, 5, 1],
 *  the method builds and returns the array [7, 2, 1, 5].  */
public static int[] buildSet (int[] arr) {
   // Write your code here:
```

Answer question 6 or question 7 (answer one question only)

6. (25 points) The method `union` returns the union of two sets. The union of two sets is the set containing all the elements of the first set and all the elements of the second set, without repetitions. For example, if the two arrays representing the sets are [2, 3, 5] and [5, 7, 2, 6], then the method returns the array [5, 7, 2, 6, 3] (not necessarily in this order). Write the method.

```
/** Returns the union of the two given sets. */
public static int[] union (int[] set1, int[] set2) {
    // Write your code here:
```

7. (25 points) The method `intersection` returns the intersection of two sets. The intersection of two sets is the set containing all the elements common to both sets. For example, if the two arrays representing the sets are [2, 3, 5] and [5, 7, 2, 6] then the method returns the array [2, 5]. Write the method.

```
/** Returns the intersection of two sets. */
public static int[] intersection (int[] set1, int[] set2) {
    // Write your code here:
```

(End of Exam, Help pages follow)

The Sets class (this page is supplied together with the exam)

```
/** This class features operations on sets that contain integer values.
 *  A set is a collection of values without repetition.
 *  For example, 2, 5, 2, 7 is not a set. {2, 5, 7} is a set.
 *  The order of the elements in the set is insignificant.
 *  For example, {2, 5, 7} is the same set as {5, 7, 2}.
 *
 *  In this implementation sets are represented as arrays of integers.
 *  In particular, a set that contains N elements is implemented by an
 *  int[] array of size N. */

public class Sets {

    public static void main(String[] args){
        // Creates two sets using the buildSet function:
        int[] a1 = { 2, 5, 3, 2, 7, 5 };
        int[] s1 = buildSet(a1);
        System.out.print("s1 = ");
        print(s1);

        int[] a2 = { 7, 5, 5, 6, 7 };
        int[] s2 = buildSet(a2);
        System.out.print("s2 = ");
        print(s2);

        // Tests the elementOf function:
        System.out.println("5 is an element of s1: " + elementOf(5, s1));
        System.out.println("8 is an element of s1: " + elementOf(8, s1));

        // Prints the intersection of s1 and s2:
        System.out.print("Intersection of s1 and s2 = ");
        print(intersection(s1,s2));

        // Prints the union of s1 and s2:
        System.out.print("Union of s1 and s2 = ");
        print(union(s1, s2));

        // Tests the random function:
        System.out.println("A random value drawn from the set s1: " + random(s1));
    }

    // The class functions (that you have to implement) are listed in next page.
}
```

Example of the program compilation and execution:

```
% javac Sets.java
% java Sets
s1 = { 2 5 3 7 }
s2 = { 7 5 6 }
5 is an element of s1: true
8 is an element of s1: false
Intersection of s1 and s2 = { 5 7 }
Union of s1 and s2 = { 2 5 3 7 6 }
A random value drawn from the set s1: 5
```

The Sets class, continued (this page is supplied together with the exam)

```
public class Sets {

    public static void main(String[] args) {
        // Same main method as in previous page
    }

    /** Prints the elements of the given set, starting and ending with curly brackets.
     *  For example, if the array contains the elements 7,2,1,5, prints { 7, 2, 1, 5 }  */
    public static void print(int[] set) {
    }

    /** Checks if the given element exists in the given set. */
    public static boolean elementOf(int e, int[] set) {
    }

    /** Returns a random element chosen from the given set. */
    public static int random (int[] set) {
    }

    /** Builds a set from a given array.
     *  For example, if the given array is [7, 2, 7, 1, 5, 1],
     *  the method builds and returns the array [7, 2, 1, 5].  */
    public static int[] buildSet (int[] arr) {
    }

    /** Returns the union of the two given sets. */
    public static int[] union (int[] set1, int[] set2) {
    }

    /** Returns the intersection of the two given sets. */
    public static int[] intersection (int[] set1, int[] set2) {
    }

}
```

Math Library API (this page is supplied together with the exam)

If you need to use a mathematical function, you may find this page helpful (it contains many more functions than you actually need for this exam).

`static int`	**`abs`**`(int a)` Returns the absolute value of an int value.
`static double`	**`log`**`(double a)` Returns the natural logarithm (base *e*) of a double value.
`static double`	**`log10`**`(double a)` Returns the base 10 logarithm of a double value.
`static int`	**`max`**`(int a, int b)` Returns the greater of two int values.
`static int`	**`min`**`(int a, int b)` Returns the smaller of two int values.
`static double`	**`pow`**`(double a, double b)` Returns the value of the first argument raised to the power of the second argument.
`static double`	**`random`**`()` Returns a double value, greater than or equal to 0.0 and less than 1.0.
`static long`	**`round`**`(double a)` Returns the closest long to the argument, with ties rounding up.
`static double`	**`sqrt`**`(double a)` Returns the correctly rounded positive square root of a double value.