Booklet no:_____

IDC Herzliya **ID:** _____

Efi Arazi School of Computer Science

Introduction to Computer Science

Final Examination

2018, Moed Bet

- The exam lasts 3 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.
- Write all your answers on the front pages only; don't write at the back of the pages.
- 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, you may 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 English or in 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 parameters, there is no need to validate the parameters 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 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.
- All the materials and documentation that you need for the exam is given to you in the help pages that accompany the exam.
- If you wish to separate / unstaple the help pages you can do so.
- No points will be taken for trivial syntax errors.

Its forbiden to answer
on the other side of the page!

Good Luck!

Questions 1-15 are about the `Set` and `SetDemo` classes, described in help pages 1 and 2. You must spend about ten minutes reading these help pages *now*, before you start answering any of the questions below.

Most of questions 1-15 deal with implementing the methods of the `Set` class. When implementing any such method, assume that all the other methods in the class have been implemented correctly, and you can use them as you please. The more you'll use other `Set` methods, the better will be your implementation.

1. (4 points) Implement the `contains` method.

/** Returns true if this set contains the given value (e), false otherwise. */

```
public boolean contains(int e) {
   // Write your code here:
```

2. (6 points) Implement the `add` method. Note: if adding the given value will cause the `elements` array to reach capacity, the method should multiply the size of the array. Tip: use other `Set` methods as much as you can. This general tip is relevant to other questions in this exam, and we will not repeat it from now on.

/** Adds the given value (e) to this set.
* If this set already contains the given value, does nothing. /*

```
public void add(int e) {
   // Write your code here:
```

3. (6 points) Implement the following constructor, which creates a set (an object of type `Set`) from the values of the given array (`arr`).

/** Constructs a set of integers from the given array. */

```
public Set(int[] arr) {
   // Write your code here:
```

4. (6 points) Implement the `toString` method. You can build a `String`, or use `StringBuilder`, we don't care. Note that each element in the string is followed by a comma (,), except for the last element, which is followed by a right curly parentheses (}).

```
/** Returns a string representing this set, in the form of "{e1, e2, e3, ...}",
 *  where the e's are the set elements. */
public String toString() {
   // Write your code here:
```

5. (8 points) Implement the `intersection` method. Note: if you want, you can access the *i*-th element of the `other` object using `other.elements[i]`. In general, each method in the `Set` class can access every private field of every object of type `Set`. This general note is relevant also to other questions in the exam, and we will not repeat it from now on.

/** Returns the intersection of this set and the other set. The intersection of sets s1 and s2
 * is a set containing all the elements that are both in s1 and in s2. */

```
public Set intersection(Set other) {
   // Write your code here:
```

Answer one of questions 6 or 7 (answer 1 question only)

6. (6 points) Implement the `union` method.

```
/** Returns the union of this set and the other set. The union of sets s1 and s2
 *  is a set containing all the elements that are either in s1, or in s2, or in both sets. */
public Set union(Set other) {
   // Write your code here:
```

7. (6 points) Implement the `subsetOf` method.

```
/** Returns true if this set is a subset of the other set.
 *  Set s1 is a subset of set s2 if every element of s1 is also an element in s2.
 *  Note: the empty set is a subset of any set. */
public boolean subsetOf(Set other) {
   // Write your code here:
```

8. (6 points) Implement the `diff` method.

/** Returns the difference between this set and the other set. The difference between sets s1 and s2
 * is a set containing all the elements that are in s1 but not in s2. */

```
public Set diff(Set other) {
   // Write your code here:
```

9. (6 points) Implement the `equals` method. For maximum points in this question, your solution should not use the `elements` array directly.

```
/** Returns true if this set equals the other set.
 *  Set s1 equals set s2 if both sets contain exactly the same elements. */
public boolean equals(Set other) {
   // Write your code here:
```

10. (6 points) Assume that the `Set` class does not include any implementation of the `equals` method. Consider the following client code, that can appear in any class (like `SetDemo`):

```
int[] arr5 = {7, 9, 3};
Set s5 = new Set(arr5);

int[] arr6 = {7, 9, 3};
Set s6 = new Set(arr6);

System.out.println(s5.equals(s6));
```

(a) Describe what would will happen when this code will be executed.
(b) Explain your answer.

11. (8 points) Implement the `remove` method. For maximum points in this question, your solution must be as efficient as possible: it should not call any other method, and it should require at most N processing steps, where N is the current size of this set.

```
/** Removes the given value (e) from this set.
 * If this set does not contain the given value, does nothing. */
public void remove(int e) {
   // Write your code here:
```

12. (4 points) Implement the static method `symmetricDiff` of the `SetDemo` class. Note: since this method is not in the `Set` class, it cannot access the private fields of any of the two sets.

```
/** Returns the symmetric difference between the two given sets.
 *  The symmetric difference between sets s1 and s2 is a set containing all the elements that
 *  are in s1 but not in s2, and all the elements that are in s2 but not in s1. */
public static Set symmetricDiff(Set s1, Set s2) {
   // Write your code here:
```

13. (4 points) Explain why the `resize` method of the `Set` class is `private`, and not `public`.

14. (14 points) Assume a different implementation of the `Set` class, in which the `elements` array is always kept sorted in increasing order. In this setting, we can write a fast implementation of the `contains` method, which uses *binary search*. For maximum points, the implementation should be recursive. Note: if you write a recursive implementation, you can write an additional, private method.

/** Returns true if this set contains the given value (e), false otherwise. */

```
public boolean contains(int e) {
   // Write your code here:
```

15. (6 points) The current implementation of the `Set` class uses an array to store the set elements. Suppose that instead of using an array, we will use a linked list for storing the set elements. Answer the following three questions: (a) The fact that we use a linked list for storing the set elements should be documented in the `Set` class API. True? False? explain your answer. (b) Describe one benefit of using a linked list for storing the set elements, compared to using an array. (c) Describe one limitation of using a linked list for storing the set elements, compared to using an array.

The last question in the exam (question 16) has nothing to do with sets.

16. (10 points) Help page 3 describes matrix multiplication. Implement the following method.

```
/** Returns the product of the two given matrices of int values.
 *  Each matrix is represented by a rectangular 2D array.
 *  If the given matrices are not product-compatible, throws a runtime exception. */
public static int[][] mult(int[][] a, int[][] b) {
   // Write your code here:
```

The Set Class (help page 1)

```
/** The Set class represents a set of integers, of unlimited size. A set is a collection of elements with neither
 *  repetition nor order. For example, {3, 1, 2, 1} is not a valid set, since 1 appears more than once.
 *  The valid version of this set would be {3, 1, 2}, or {1, 2, 3}, or any other arrangement of the set
 *  elements. Since the order of elements in a set is insignificant, all these sets are considered equal. */

public class Set {
   private static final int DEFAULT_CAPACITY = 100;  // Default capacity of this set.
   private int[] elements;                           // The elements of this set, in no particular order.
   // The number of elements in this set. For example, the size of {5, 1, 2, 9} is 4, and the size of the empty set {} is 0.
   private int size;

   /** Constructs an empty set. */
   public Set() {
      elements = new int[DEFAULT_CAPACITY];  // Constructs an array of set elements, with the default capacity.
      size = 0;                              // Sets the size of the new set to 0.
   }

   /** Constructs a set of integers from the given array. */
   public Set(int[] arr) {}

   /** Returns the size of this set. */
   public int size() { return size; }

   /** Returns true if this set contains the given value, false otherwise. */
   public boolean contains(int e) {}

   /** Adds the given value to this set. If this set already contains the given value, does nothing. /*
   public void add(int e) {}

   /** Removes the given value from this set. If this set does not contain the given value, does nothing. */
   public void remove(int e) {}

   /** Returns the intersection of this set and the other set. The intersection of sets s1 and s2
    *  is a set containing all the elements that are both in s1 and in s2. */
   public Set intersection(Set other) {}

   /** Returns the union of this set and the other set. The union of sets s1 and s2
    *  is a set containing all the elements that are either in s1, or in s2, or in both sets. */
   public Set union(Set other) {}

   /** Returns true if this set is a subset of the other set.
    *  Set s1 is a subset of set s2 if every element of s1 is also an element in s2.
    *  Note: the empty set is a subset of any set.
   public boolean subsetOf(Set other) {}

   /** Returns the difference between this set and the other set.
    *  The difference between sets s1 and s2 is a set containing all the elements that are in s1 but not in s2. */
   public Set diff(Set other) {}

   /** Returns true if this set equals the other set.
    *  Set s1 equals set s2 if both sets contain exactly the same elements. */
   public boolean equals(Set other) {}

   /** Returns a string representing this set in the form of {e1, e2, e3, ...} where the e's are the set elements. */
   public String toString() {}

   // Resizes the array that represents this set, by doubling its capacity.
   private void resize() {
      int[] temp = new int[2 * elements.length];
      System.arraycopy(elements, 0, temp, 0, elements.length); // copies the elements array to the temp array
      elements = temp;
   }

} // End of the Set class
```

The SetDemo Class (help page 2)

/** This class demonstrates how to use the services of the Set class. */

```
public class SetDemo {

   public static void main(String args[]) {
      int[] arr1 = {4, 1, 2};
      Set s1 = new Set(arr1);
      s1.add(1);
      s1.add(3);
      System.out.println(s1);                    // Prints {4, 1, 2, 3}

      int[] arr2 = {3, 6, 2};
      Set s2 = new Set(arr2);
      System.out.println(s2);                    // Prints {3, 6, 2}

      System.out.println(s1.intersection(s2));   // Prints {2, 3}
      System.out.println(s1.union(s2));          // Prints {4, 1, 2, 3, 6}
      System.out.println(s1.diff(s2));           // Prints {4, 1}

      int[] arr3 = {2, 3, 6};
      Set s3 = new Set(arr3);
      System.out.println(s3.equals(s2));         // Prints true

      // The code below this point is relevant only to question 12, so you can read it later.

      System.out.println(symmetricDiff(s1,s2)); // Prints {4, 1, 6}

   }


   /** Returns the symmetric difference between the two given sets.
    * The symmetric difference between sets s1 and s2 is a set containing all the elements that
    * are in s1 but not in s2, and all the elements that are in s2 but not in s1. */
   public static Set symmetricDiff(Set s1, Set s2) {
      // You have to write this code in question 12
   }

} // end of the SetDemo class
```

Matrix Multiplication (help page 3)

We say that two matrices A and B are "product-compatible" if the number of columns in matrix A equals the number of rows in matrix B.

The multiplication (also called "product") of two product-compatible matrices C = A x B is defined as follows:

Each entry $C_{i,j}$ in matrix C is set to the inner product of row i of matrix A and column j of matrix B.

Here is an example of multiplying two specific matrices:

$$\begin{bmatrix} 3 & 12 & 4 \\ 5 & 6 & 8 \\ 1 & 0 & 2 \end{bmatrix} \begin{bmatrix} 7 & 3 & 8 \\ 11 & 9 & 5 \\ 6 & 8 & 4 \end{bmatrix}$$

$$= \begin{bmatrix} 3*7+12*11+4*6 & 3*3+12*9+4*8 & 3*8+12*5+4*4 \\ 5*7+6*11+8*6 & 5*3+6*9+8*8 & 5*8+6*5+8*4 \\ 1*7+0*11+2*6 & 1*3+0*9+2*8 & 1*8+0*5+2*4 \end{bmatrix}$$

$$= \begin{bmatrix} 177 & 149 & 100 \\ 149 & 133 & 102 \\ 19 & 19 & 16 \end{bmatrix}$$

In this particular example, the two matrices happen to be *square*, having the same number of rows and columns. In general, in order for the multiplication operation to be valid, we only require that the two matrices will be product-compatible.