YOU MAY READ THIS PAGE BEFORE THE EXAM BEGINS

1. You have 75 minutes to create and submit the 2 programs in this exam.

2. When instructed to start, download the following IntelliJ project containing the starter files for the exam problems (use student for username and enigma for password):

   https://www.cs.umb.edu/~siyer/teaching/restricted/cs210_sample_programming_exam2.zip

3. You may use the text, your notes, your code from the projects, and the code on the CS210 course website. No form of communication is permitted (eg, talking, texting, etc.) during the exam, except with the course staff.

4. Submit your work on Gradescope under the assignment named "Programming Exam 2 (Section XYZ)", where XYZ denotes your discussion section number.

5. Return this exam sheet to the course staff with your name written at the top. Failing to do so will void your exam submission on Gradescope.

6. Your programs will be graded based on correctness, clarity (including comments), design, and efficiency.

7. Discussing the exam contents with anyone who has not taken the exam is a violation of the academic honesty code.

DO NOT READ FURTHER OR DOWNLOAD THE STARTER FILES UNTIL SO INSTRUCTED
Problem 1. (18 Points) Design an efficient data type called ThreadedSet to store a *threaded set of strings*, which maintains a set of strings (no duplicates) and the order in which the strings were inserted, according to the following API:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ThreadedSet()</td>
<td>creates an empty threaded set</td>
</tr>
<tr>
<td>void add(String s)</td>
<td>adds s to this set if it is not already in the set</td>
</tr>
<tr>
<td>boolean contains(String s)</td>
<td>returns true if s is in this set, and false otherwise</td>
</tr>
<tr>
<td>String previousKey(String s)</td>
<td>returns the string that was added to this set immediately before s; returns null if s is the first string added; and throws java.util.NoSuchElementException if s is not in this set</td>
</tr>
</tbody>
</table>

```java
javac -d out src/ThreadedSet.java
t mkdir ThreadedSet
Adding aardvark, bear, cat, and bear to a threaded set...
set.contains(bear) = true
set.contains(tiger) = false
set.previousKey(cat) = bear
set.previousKey(bear) = aardvark
set.previousKey(aardvark) = null
```

Problem 2. (7 Points) Implement the function private static void commonString(String[] a, String[] b) in CommonString.java such that it returns a string that appears in both a and b, or null if the arrays don’t have any strings in common.

```java
javac -d out src/CommonString.java
t mkdir CommonString
commonString(a, b) = GCA
commonString(a, c) = null
```

Files to Submit

1. ThreadedSet.java
2. CommonString.java
Answers

Problem 1.

ThreadedSet.java

```java
import dsa.SeparateChainingHashST;
import java.util.NoSuchElementException;
import stdlib.StdOut;

// A data type to store a threaded set of strings, which maintains a set of strings (no duplicates) and the order in which the strings were inserted.
public class ThreadedSet {
    private String prev;
    private SeparateChainingHashST<String, String> st;

    // Constructs an empty threaded set.
    public ThreadedSet() {
        prev = "__null__";
        st = new SeparateChainingHashST<String, String>();
    }

    // Adds s to this set if it is not already in the set.
    public void add(String s) {
        if (!contains(s)) {
            st.put(s, prev);
            prev = s;
        }
    }

    // Returns true if this set contains s, and false otherwise.
    public boolean contains(String s) {
        return st.contains(s);
    }

    // Returns the string that was added to this set immediately before s; returns null if s is the first string added; and throws java.util.NoSuchElementException if s is not in this set.
    public String previousKey(String s) {
        if (!contains(s)) {
            throw new NoSuchElementException();
        }
        String value = st.get(s);
        return value == "__null__" ? null : value;
    }

    // Unit tests the data type [DO NOT EDIT].
    public static void main(String[] s) {
        ThreadedSet set = new ThreadedSet();
        StdOut.println("Adding aardvark, bear, cat, and bear to a threaded set...\n");
        set.add("aardvark");
        set.add("bear");
        set.add("cat");
        set.add("bear");
        StdOut.println("set.contains(bear) = " + set.contains("bear"));
        StdOut.println("set.contains(tiger) = " + set.contains("tiger"));
        StdOut.println("set.previousKey(cat) = " + set.previousKey("cat"));
        StdOut.println("set.previousKey(bear) = " + set.previousKey("bear"));
        StdOut.println("set.previousKey(aardvark) = " + set.previousKey("aardvark"));
    }
}
```

Problem 2.

CommonString.java

```java
import dsa.SeparateChainingHashST;
import java.util.Arrays;
import stdlib.StdOut;

public class CommonString {
    // Returns a string that appears both in a and b, or null if the arrays don't have any strings in common.
    private static String commonString(String[] a, String[] b) {
        SeparateChainingHashST<String, String> st = new SeparateChainingHashST<String, String>();
        
        // ... (similar code to ThreadedSet)...
    }
}
```
for (String s : b) {
    st.put(s, s);
}

for (String s : a) {
    if (st.contains(s)) {
        return s;
    }
}
return null;

// Entry point [DO NOT EDIT].
public static void main(String[] args) {
    String a = "GCA TCA ACG ACT GTC AGC GTA ATG";
    String b = "GAT GCA CAG GCT TCG GTC CTA ATG";
    String c = "it was the best of times it was the worst of times";
    String[] aList = a.split("\s+");
    String[] bList = b.split("\s+");
    String[] cList = c.split("\s+");
    StdOut.println("a = " + Arrays.toString(aList));
    StdOut.println("b = " + Arrays.toString(bList));
    StdOut.println("c = " + Arrays.toString(cList));
    StdOut.println("commonString(a, b) = " + commonString(aList, bList));
    StdOut.println("commonString(a, c) = " + commonString(aList, cList));
}