Name:

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 and extract the following IntelliJ Project under the `~/workspace` folder if you do not have it there already.

   `https://www.cs.umb.edu/~siyer/teaching/dsaj.zip`

3. Open the project in IntelliJ. To create a program, right-click on the `src` folder under `dsaj` in the top-left window and then select the *New* → *File* menu. Enter the name of the program in the pop-up window. Note that the name is case-sensitive and must match the suggested name exactly.
4. You may use the text, your notes, your code from the assignments, 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.
5. Submit your programs (`.java` files) on Gradescope under the assignment named `Sample Programming Exam 2`.
6. 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.
7. You are *not* allowed to leave the exam hall before the official end time even if you are done early.
8. Your programs will be graded based on correctness, clarity, and efficiency.
9. Discussing the exam contents with anyone who has not taken the exam is a violation of the academic honesty code.

DO NOT READ FURTHER 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:

ThreadedSet

`ThreadedSet()`	creates an empty threaded set
`void add(String s)`	adds `s` to this set if it is not already in the set
`boolean contains(String s)`	returns `true` if `s` is in this set, and `false` otherwise
`String previousKey(String 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

Enter the following starter code in the program and replace the ellipsis (...) with your code.

ThreadedSet.java

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

public class ThreadedSet {
    ...

    public ThreadedSet() { ... }

    public void add(String s) { ... }

    public boolean contains(String s) { ... }

    public String previousKey(String s) { ... }

    public static void main(String[] s) {
        ThreadedSet set = new ThreadedSet();
        set.add("aardvark");
        set.add("bear");
        set.add("cat");
        set.add("bear");
        StdOut.println(set.contains("bear"));
        StdOut.println(set.contains("tiger"));
        StdOut.println(set.previousKey("cat"));
        StdOut.println(set.previousKey("bear"));
        StdOut.println(set.previousKey("aardvark"));
    }
}
```

>_ ~/workspace/dummy_project

```
$ javac -d out src/ThreadedSet.java
$ java ThreadedSet
true
false
bear
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.

Enter the following starter code in the program and replace the ellipsis (...) with your code.

CommonString.java

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

public class CommonString {
    private static String commonString(String[] a, String[] b) { ... }

    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(commonString(aList, bList));
        StdOut.println(commonString(aList, cList));
    }
}
```

```
>_ ~/workspace/dummy_project
$ javac -d out src/CommonString.java
$ java CommonString
GCA
null
```

Files to Submit

1. `ThreadedSet.java`
2. `CommonString.java`

Answers

Problem 1.

ThreadedSet.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();
        set.add("aardvark");
        set.add("bear");
        set.add("cat");
        set.add("bear");
        StdOut.println(set.contains("bear"));
        StdOut.println(set.contains("tiger"));
        StdOut.println(set.previousKey("cat"));
        StdOut.println(set.previousKey("bear"));
        StdOut.println(set.previousKey("aardvark"));
    }
}
```

Problem 2.

CommonString.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>();
        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(commonString(aList, bList));
        StdOut.println(commonString(aList, cList));
    }
}
```