Practice exam for CMSC131-04, Fall 2017

Q1 makePalindrome

- `Relevant topics: arrays, loops`

Write a method `makePalidrome` that takes an `int` array, return a new `int` array that contains the values from the original array in their original order, followed by the elements from the original array in reversed order.

For example, given [5, 4, 8], it will return the array [5, 4, 8, 8, 4, 5]. The original array should not be modified.

Q2 countPairs

- `Relevant topics: recursion, strings`
- `Note: From codingbat-1`

Write a recursive method `countPairs` that takes a String and returns the number of pairs in the string, where we'll define a "pair" in a string is two instances of a char separated by a char. So "AxA" the A's make a pair. Pair's can overlap, so "AxAxA" contains 3 pairs -- 2 for A and 1 for x. Recursively compute the number of pairs in the given string. Do not use any looping constructs or string methods other than `length`, `charAt` and `substring`.

Other example test cases:

- countPairs("axa") → 1
- countPairs("axax") → 2
- countPairs("axbx") → 1

Please complete the following about your implementation:

- Did you avoid using any loops (e.g., while loops or for loops)?
  - ☐ yes
  - ☐ I didn't read the instructions and am going to lose lots of points
- What is the base case?
- What are the recursive calls you make?
- In what way are the arguments to the recursive calls simpler?

Your solution (use extra paper if needed; this is only a practice exam)

Q3 selectStrings

- `Relevant topics:` List operations, iteration, String methods

Write a static method `selectStrings` that takes two arguments:

- A list of Strings
- A string prefix

The `selectStrings` method returns a list of Strings containing just the strings from the list provided as an argument that start with the provided prefix. The arguments should not be modified. For example, given the arguments [“CMSC132”, “MATH140”, “PSYC200”, “CMSC250”] and “CMSC”, it should return [“CMSC132”, “CMSC250”].

Q4 Exception test from Lab on Dec 11th

- `Relevant topics:` Exceptions, try-catch-finally

Consider the following code:

```
public static void main(String args[]) {
  try {
    int [] a = {-3, 5, 10};

    System.out.println(checkPositive(a, 0));
    System.out.println(checkPositive(a, 0));
    System.out.println(checkPositive(a, 0));
    System.out.println(checkPositive(a, 0));
    System.out.println(checkPositive(a, 0));

  } catch (NullPointerException e) {
    System.out.println("Null pointer exception");
  } catch (ArrayIndexOutOfBoundsException e) {
    System.out.println("Array index out of bounds exception");
  } finally {
    System.out.println("finished");
  }
}

private static boolean checkPositive(int[] a, int i) {
  try {
    return a[i] > 0;
  } catch (ArrayIndexOutOfBoundsException e) {
    System.out.println("index " + i + " is out of bounds");
    return false;
  } finally {
    System.out.println("done");
  }
}
```

Changed arguments for the 5 calls to checkPositive so that the output of this program is as shown.

Expected output	Which call	Arguments to checkPositive call
done true done false index 5 is out of bounds done false done Null pointer exception finished	1st	
	2nd	
	3rd	
	4th	
	5th	

Q5 ints/double

- `Relevant topics:` integer vs floating point math, conversion between int and double

What does this code print?

```
double z = 3.9;
System.out.println(z);
int q = (int)z;
System.out.println(q);
System.out.println(q / 2 * z);
double w = 505 / 10+0.1;
System.out.println(w);
```

Print statement	Output
1st println	
2nd println	
3rd println	
4th println	

Q6 Drawing Triangle

- `Relevant topics:` Loops, Nested loops, Manually throwing an exception

Write a method `drawTriangle` that will print out a right triangle whose length (height and width) is represented by the integer parameter. Look at the examples for what your code should do. If the length is less than zero, throw an IllegalArgumentException.

`drawTriangle(4);`	`*`
	`**`
	`***`
	`****`

drawTriangle(1);	`*`

drawTriangle(5);	`*`
	`**`
	`***`
	`****`
	`*****`

drawTriangle(-1);	*IllegalArgumentException*.

Q7 Design of class for `Course`

See description of Q7 on page 11

Provide just the constructors for course here; other methods on next page

Q7 Provide your implementations of addSection, equals and allIds() here

Q8 recursion mystery

Consider the following function.

```
static int mystery(int x) {
    if (x == 0)
      return 0;
    return x%10 + mystery(x/10);
  }
```

Show the calls to mystery in the order they are made. For each call, show the argument to mystery and the value returned. Note: there are more rows than you will need.

Calls	Values returned
mystery(567)	

Q7 Design of class for Course (tear this page off if desired)

- `Relevant topics: equals method, copy constructor, deep copy, ArrayLists`

This is a simplified version of the course class from project 10. For purposes of this question, the Course class has just three instance variables:

```
public class Course {
  private String courseId;
  private String name;
  private ArrayList<Section> sections;
 }
```

You can assume that appropriate getters are provided for those three instance variables.
You need to write the following:

- A constructor that takes two arguments, a `courseId` and a `name`, and initializes sections to be empty.
- A copy constructor that performs a deep copy. Assume that the class `Section` has a copy constructor that performs a deep copy.
- An instance method addSection that takes one Section as an argument and adds it to the `sections` for the course.
- An appropriate equals method such that two `Course`'s are equal if and only if their `courseId`'s, `name`'s and `section`'s are equal.
    - You do not need to implement `hashCode`
- A static method `allIds()` that takes no arguments and returns a `TreeSet<String>` containing all of the `courseIds` that were ever created. Define any static variables you need to support this.

Reference for practice final, CMSC 131-04

The following includes all of the methods you will need to use for these questions, as well as some you won't need.

```
String methods
int length();
boolean startsWith(String s);
boolean substring(int beginIndex, int endIndex);
char charAt(int i);

List<T> methods
int size();
boolean add(T t);
boolean contains(T t);
boolean remove(T t);
boolean removeAll(Collection<T> t);
T get(int i);
T set(int i, T value);

Set<T> methods
int size();
boolean add(T t);
boolean contains(T t);
boolean remove(T t);
boolean removeAll(Collection<T> t);

Map<K,V> methods
int size();
V get(K key);

// get the value with this key if it exists, if it doesn't return the defaultValue
V getOrDefault(K key, V defaultValue);

boolean containsKey(K k);
V put(K key, V value);
V remove(K key);

// return a set of the keys in the map
Set<K> keySet();
```

You can iterate through anything `iterable`, including an array, a `List` or a `Set,` using a "for each" loop. For example, if `names` is declared as a `Set<String>`, you can use the following code to print out all the Strings in that set:

```
for(String name : names) {
  System.out.println(name);
 }
```