**Exam 1**

Name: \_\_\_\_\_\_\_\_\_\_\_

**This is an open book exam. You can look at the book and use your favorite IDEs. But you cannot consult with others. You must submit the work within the class time.**

**Part I: Write Program**

1. (15 pts) Please submit your code to LiveLab (URL: livelab.georgiasouthern.edu/JavaLiveLab2020) under PExam1a

Write a recursive method that computes the sum of the digits in an integer. Use the following method header:

**public static int** sumDigits(**long** n)

For example, **sumDigits(234)** returns 

Write a test program that prompts the user to enter an integer and displays its sum.

Hint:

sumDigits(n) = 0 if n = 0

sumDigits(n) = sumDigits(n / 10) + n % 10, if n > 0

<Sample output>

Enter an integer: 345

The sum of digits in 345 is 12

</Sample output>

2. (15 pts) Please submit your code to LiveLab (URL: livelab.georgiasouthern.edu/JavaLiveLab2020) under PExam1b

Write a program that reads five strings and displays the strings in increasing order of their last character. Strings are separated by spaces. You must use the Comparator interface with the lambda expression to specify a sort criterion.

<Sample output>

Enter five strings: Macon Savannah Atlanta Richmond Chatham

The strings in increasing order of their last character are

Atlanta Richmond Savannah Chatham Macon

</Sample output>

**Part II: Multiple-Choice**

Please write your answers on this document first. After you finish the entire test, enter your choices online to LiveLab. Log in and click Take Instructor Assigned Quiz. Choose Quiz1. You have 5 minutes to enter and submit the answers.) LiveLab URL: livelab.georgiasouthern.edu/JavaLiveLab2020

1. Fill in the code to complete the following method for checking whether a string is a palindrome.

public static boolean isPalindrome(String s) {

if (s.length() <= 1) // Base case

return true;

else if \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

return false;

else

return isPalindrome(s.substring(1, s.length() - 1));

}

A. (s.charAt(1) != s.charAt(s.length() - 1)) // Base case

B. (s.charAt(0) != s.charAt(s.length() - 1)) // Base case

C. (s.charAt(0) != s.charAt(s.length())) // Base case

D. (s.charAt(1) != s.charAt(s.length())) // Base case

#

2. Fill in the code to complete the following method for binary search.

public static int recursiveBinarySearch(int[] list, int key) {

int low = 0;

int high = list.length - 1;

return \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

}

public static int recursiveBinarySearch(int[] list, int key,

int low, int high) {

if (low > high) // The list has been exhausted without a match

return -low - 1; // Return -insertion point - 1

int mid = (low + high) / 2;

if (key < list[mid])

return recursiveBinarySearch(list, key, low, mid - 1);

else if (key == list[mid])

return mid;

else

return recursiveBinarySearch(list, key, mid + 1, high);

}

A. recursiveBinarySearch(list, key, low, high)

B. recursiveBinarySearch(list, key, low + 1, high - 1)

C. recursiveBinarySearch(list, key)

D. recursiveBinarySearch(list, key, low - 1, high + 1)

#

3. Fill in the code to complete the following method for computing factorial.

/\*\* Return the factorial for a specified index \*/

public static long factorial(int n) {

if (n == 0) // Base case

return 1;

else

return \_\_\_\_\_\_\_\_\_\_\_\_\_; // Recursive call

}

A. n \* (n - 1)

B. factorial(n - 1) \* n

C. factorial(n - 1)

D. n

#

4. Analyze the following code:

public class Test {

public static void main(String[] args) {

int[] x = {1, 2, 3, 4, 5};

xMethod(x, 5);

}

public static void xMethod(int[] x, int length) {

System.out.print(" " + x[length - 1]);

xMethod(x, length - 1);

}

}

A. The program displays 5 4 3 2 1.

B. The program displays 1 2 3 4 5 and then raises an ArrayIndexOutOfBoundsException.

C. The program displays 1 2 3 4 6.

D. The program displays 5 4 3 2 1 and then raises an ArrayIndexOutOfBoundsException.

#

5. What are the base cases in the following recursive method?

public static void xMethod(int n) {

if (n > 0) {

System.out.print(n % 10);

xMethod(n / 10);

}

}

A. n < 0

B. no base cases

C. n > 0

D. n <= 0

#

6. To declare a class named A with a generic type, use

A. public class A(E, F) { ... }

B. public class A<E> { ... }

C. public class A(E) { ... }

D. public class A<E, F> { ... }

#

7. To create a list to store integers, use

A. ArrayList<Number> list = new ArrayList<>();

B. ArrayList<int> list = new ArrayList<int>();

C. ArrayList<Object> list = new ArrayList<>();

D. ArrayList<Integer> list = new ArrayList<>();

#

8. Which of the following declarations use raw type?

A. ArrayList<String> list = new ArrayList<>();

B. ArrayList<Object> list = new ArrayList<>();

C. ArrayList<Integer> list = new ArrayList<>();

D. ArrayList list = new ArrayList();

#

9. If you use the javac -Xlint:unchecked command to compile a program that contains raw type, what would the compiler do?

A. report compile error

B. no error and generate a class file

C. report warning without generating a class file

D. report warning and generate a class file

#

10. To declare an interface named A with two generic types, use

A. public interface A(E) { ... }

B. public interface A<E, F> { ... }

C. public interface A<E> { ... }

D. public interface A(E, F) { ... }

#

11. What is the output of the following code?

import java.util.\*;

public class Test {

public static void main(String[] args) {

List<String> list1 = new ArrayList<>();

list1.add("Atlanta");

list1.add("Macon");

list1.add("Savanna");

List<String> list2 = new ArrayList<>();

list2.add("Atlanta");

list2.add("Macon");

list2.add("Savanna");

List<String> list3 = new ArrayList<>();

list3.add("Macon");

list3.add("Savanna");

list3.add("Atlanta");

System.out.println(list1.equals(list2) + " " + list1.equals(list3));

}

}

A. false false

B. true true

C. false true

D. true false

#

12. All the concrete classes in the Java Collections Framework implement \_\_\_\_\_\_\_\_\_\_\_\_\_.

A. the Comparator interface

B. the Comparable interface

C. the Cloneable interface

D. the Serializable interfaces

#

13. Suppose a list contains {"red", "green", "red", "green"}. What is the list after the following code?

String element = "red";

for (int i = list.size() - 1; i >= 0; i--)

if (list.get(i).equals(element))

list.remove(element);

A. {}

B. {"red", "green"}

C. {"green"}

D. {"red", "red", "green"}

E. {"green", "green"}

#

14. To find a maximum object in an array of strings (e.g., String[] names = {"red", "green", "blue"}), use

A. Arrays.sort(names)

B. Collections.max(Arrays.asList(names))

C. Arrays.max(names)

D. Collections.max(names)

E. None of the above

#

15. Suppose list list1 is [1, 2, 5] and list list2 is [2, 3, 6]. After list1.addAll(list2), list1 is \_\_\_\_\_\_\_\_\_\_.

A. [2]

B. [1, 5]

C. [1, 2, 2, 3, 5, 6]

D. [1, 2, 3, 5, 6]

#

16. What is the output of the following code?

import java.util.\*;

public class TestLinkedHashSet {

public static void main(String[] args) {

// Create a linked hash set

Set<String> set = new LinkedHashSet<>();

// Add strings to the set

set.add("London");

set.add("Paris");

set.add("New York");

set.add("San Francisco");

set.add("Beijing");

set.add("New York");

System.out.println(set);

}

}

A. ["London", "Paris", "New York", "San Francisco", "Beijing"]

B. ["London", "Paris", "New York", "San Francisco", "Beijing", "New York"]

C. [London, Paris, New York, San Francisco, Beijing]

D. [London, Paris, New York, San Francisco, Beijing, New York]

#

17. Analyze the following code.

import java.util.\*;

public class Test {

public static void main(String[] args) throws Exception {

TreeSet<String> set = new TreeSet<>();

set.add("Red");

set.add("Yellow");

set.add("Green");

set.add("Blue");

SortedSet temp = set.headSet("Purple");

System.out.println(temp.first());

}

}

A. The program displays Yellow

B. The program displays Blue

C. The program displays Green

D. The program displays Red

E. The program displays Purple

#

18. Analyze the following code.

import java.util.\*;

public class Test {

public static void main(String[] args) throws Exception {

TreeSet<String> set = new TreeSet<>();

set.add("Red");

set.add("Green");

set.add("Blue");

System.out.println(set.last());

}

}

A. The program cannot compile, because the last() method is not defined in Set.

B. The program displays Green

C. The program may display Red, Blue, or Green.

D. The program displays Blue

E. The program displays Red

#

19. Analyze the following code:

public class Test {

public static void main(String[] args) {

Map<String, String> map = new HashMap<>();

map.put("123", "John Smith");

map.put("111", "George Smith");

map.put("123", "Steve Yao");

map.put("222", "Steve Yao");

}

}

A. After all the four entries are added to the map, "123" is a key that corresponds to the value "John Smith".

B. After all the four entries are added to the map, "John Smith" is a key that corresponds to the value "123".

C. After all the four entries are added to the map, "123" is a key that corresponds to the value "Steve Yao".

D. A runtime error occurs because two entries with the same key "123" are added to the map.

E. After all the four entries are added to the map, "Steve Yao" is a key that corresponds to the value "222".

#

20. To empty a Collection or a Map, you use the \_\_\_\_\_\_\_\_\_\_ method.

A. setEmpty

B. clear

C. zero

D. empty