|  |  |
| --- | --- |
|  | **University of Maryland College Park** |
| **Dept of Computer Science** |
| ***CMSC132 Spring 2012*** |
| ***Midterm Key*** |

*First Name (PRINT):* **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

*Last Name (PRINT):* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*University ID:* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Section/TAName:* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*I pledge on my honor that I have not given or received any unauthorized assistance on this examination.*

*Your signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

***Instructions***

* *This exam is a closed-book and closed-notes exam.*
* *Total point value is 100 point (110 Honors).*
* *The exam is a 50 minutes exam.*
* *Please use a pencil to complete the exam.*
* ***WRITE NEATLY****. If we cannot understand your answer, we will not grade it (i.e., 0 credit).*

***Grader Use Only***

|  |  |  |  |
| --- | --- | --- | --- |
| #1 | Algorithmic Complexity | (20) |  |
| #2 | Program Correctness | (8) |  |
| #3 | Hashing | (8) |  |
| #4 | Language Features | (10) |  |
| #5 | Sets and Maps | (16) |  |
| #6 | Linear Data Structures | (38) |  |
| #7 | Honors | (10) |  |
| **Total** | Total | (100/110) |  |

**Problem 1 (20 pts) Algorithmic Complexity**

1. (9 pts) Calculate the asymptotic complexity of the code snippets below (using big-O notation) with

 respect to the problem size **n**.

 **Answer:**

* 1. f(**n**) = O( **1** )

 i = 1;

 while (i < n) {

 if (i % 2 == 0) {

 break;

 }

 i = i + 1;

 }

* 1. f(**n**) = O( log(**n**) )

 for (i = 2; i < n; i \*= 2) {

 System.*out*.println(i);

 }

* 1. f(**n**) = O( **n** )

 for (int i = 0; i < n / 2; i++) {

 for (int k = n; k <= n; k++) {

 System.*out*.println("Computing");

 }

 }

1. (4 pts) Give the asymptotic bound of the following functions:

 **Answer:**

1. **n** log(**n**) + 4**n**2 + 7 f(**n**) = O( **n**2 )
2. **n**k + **n!** f(**n**) = O( **n!** )
3. (4 pts) List the following big-O expressions in order of asymptotic complexity (lowest complexity first).

 O(**n**log(**n**)) O(**nn**) O(**n**) O(**kn**)

**Answer:**

 O(**n**) O(**n**log(**n**)) O(**kn**) O(**nn**)

1. (3 pts) Indicate the complexity (big O) for an algorithm whose running time increases by a constant each time we double the input data set.

**Answer:** O(log(**n**))

**Problem 2 (8 pts) Program Correctness and Exceptions**

**NOTE: IllegalArgumentException is NOT a checked exception**

1. (5 pts) The method **evaluate** may throw an exception (IllegalArgumentException) according to the argument value provided. Modify the following code fragment so the exception is handled by a catch clause in processInfo. The catch clause will display the message “Invalid argument” using System.out.println.

 public void processInfo() {

 int y = 20;

 int result = **evaluate**(y);

 System.*out*.println(result);

 }

**Answer:**

 public void processInfo() {

 int y = 20;

 try {

 int result = evaluate(y);

 System.out.println(result);

 }

 catch(IllegalArgumentException e) {

 System.out.println("Invalid argument");

 }

 }

1. (3 pts) The method **computeValue** throws a checked exception (named MyException). Modify the following code so we don’t have to handle it in the method **generateResults**.

 public void generateResults() {

 **computeValue**(10);

}

**Answer:**

 public void generateResults() throws MyException {

 computeValue(10);

 }

**Problem 3 (8 pts) Hashing**

1. (4 pts) Does the following class satisfy the Java Hash Code contract? Explain briefly (yes or no answer will receive no credit.) The method random.nextInt() returns a random integer value.

public class Game {

 private int number;

 public static Random *random* = new Random();

 public boolean equals(Object o) {

 if (o == this)

 return true;

 if (!(o instanceof Game))

 return false;

 return ((Game)o).number == number;

 }

 public int hashCode() {

 return number + *random*.nextInt();

 }

}

**Answer:**

No. Two objects with the same number have different hashCode values as a random value is added to number.

1. (4 pts) Does the default implementation of equals and hashCode in Java satisfy the Java Hash Code contact? Explain briefly (yes or no answer will receive no credit.)

**Answer:** Yes. Usually the default hashCode returns the object address. The equals method by default compares addresses, so two objects that are equal have the same hashCode() value.

**Problem 4 (10 pts) Java Language Features**

1. (4 pts) The **Ship** interface defines a single method with the following signature: ***public void stop();*** Using an anonymous inner class complete the following assignment where **x** is assigned an object that

 implements the **Ship** interface and the method stop() will print (using System.out.println) the message

 “Stop Ship”.

**Answer:** Ship y **=** new Ship() { public void stop() { System.out.println(“Stop Ship”); }};

1. (6 pts) Make the following class generic so that it can deal with an arbitrary class rather than only Strings. Feel free to cross out parts of the following code.

public class Col {

 private ArrayList<String> c;

public String get() { return c.remove(0); }

public void insert(String value) { c.add(value); }

}

**Answer:**

public class Col<T> {

 private ArrayList<T> c;

public T get() { return c.remove(0); }

public void insert(T value) { c.add(value); }

}

**Problem 5 (16 pts) Sets and Maps**

Complete the following program. The getSortedIds method returns a sorted set of integers. The program’s output is [8, 20, 33]. You may not use Collections.sort().

**Answer:**

public class Students {

 public static void main(String[] args) {

 Map<String, Integer> namesAndIds = new TreeMap<String, Integer>();

 namesAndIds.put("John", 20);

 namesAndIds.put("Laura", 8);

 namesAndIds.put("Peter", 33);

 System.out.println(getSortedIds(namesAndIds));

 }

 public static Set<Integer> getSortedIds(Map<String, Integer> namesAndIds) {

 TreeSet<Integer> answer = new TreeSet<Integer>();

 for (String name : namesAndIds.keySet()) {

 answer.add(namesAndIds.get(name));

 }

 return answer;

 }

}

**You may find the following Map methods helpful:**

* V **get**(Object key) - Returns the value to which this map maps the specified key.
* V **put**(K key,V value) - Associates the specified value with the specified key in this map.
* Set<K> **keySet**() - Returns a set view of the keys contained in this map.
* boolean **isEmpty**() - Returns true if this map contains no key-value mappings.

**You may find the following Set methods helpful:**

* boolean **contains**(Object o) - Returns true if this set contains the specified element.
* boolean **add**(E o) - Adds the specified element to this set if it is not already present.
* V **remove**(Object key) - Removes the element from the set.
* boolean **isEmpty**() - Returns true if this set contains no element.

**Problem 6 (38 pts) Linear Data Structures**

Implement the methods below based on the following Java class definitions. You may not modify the Node class and you may not add any instance or static variables. In addition, you may not use the Java API LinkedList class.

public class LinkedList<T> {

 private class Node<V> {

 private V val;

 private Node<V> next;

 }

 private Node<T> head;

 private Comparator<T> comp;

 public LinkedList(Comparator<T> comparator) {

 head = null;

 comp = comparator;

 }

 public boolean sameValues(LinkedList<T> L) { **/\* You must implement \*/** }

 public void add(T element) { **/\* You must implement \*/** }

}

1. Implement the method **sameValues** that determines whether two lists have the same values. Two lists are considered the same if they have the same length and the same values (in the same order). Two empty lists are considered the same. For this problem:
	1. You must implement a non-recursive solution, otherwise you will not get credit.
	2. The lists can have different sizes and they can be empty.
	3. You may not add any auxiliary methods.

**Answer:**

public boolean sameValues(LinkedList<T> L) {

 Node<T> firstL = head;

 Node<T> secondL = L.head;

 while (firstL != null && secondL != null) {

 if (comp.compare(firstL.val, secondL.val) != 0) {

 return false;

 }

 firstL = firstL.next;

 secondL = secondL.next;

 }

 if (firstL == null && secondL == null) {

 return true;

 } else {

 return false;

 }

 }

}

1. Implement the method **add** which adds **element** to the list, so the list is kept sorted in ascending order. For this problem:
	1. You must implement a non-recursive solution.
	2. You may not add any auxiliary methods.

**Answer:**

This is the add method you implement for your linked list project. Because it is a method you must implement for a project, we cannot post the solution, but you are welcome to discuss your implementation with a TA.

**THERE IS ANOTHER PROBLEM ON THE BACK**

**(10 pts) THIS QUESTION IS FOR STUDENTS IN THE HONOR SECTION. ONLY STUDENTS IN THE HONOR SECTION WILL RECEIVE CREDIT FOR IT.**

1. (3 pts) What are the implications of a hashCode method that generates a unique value for every object?

**Answer:** Better hashing (we avoid collisions)

1. (4 pts) What class modifications are needed in order to make a class immutable?

**Answer:** No mutators, make fields final, make fields private, ensure no methods may be overridden

1. (3 pts) Provide an example/explanation that demonstrates that the objL = strL assignment below is illegal in Java.

ArrayList<String> strL = new ArrayList<String>();

ArrayList<Object> objL = strL; // Illegal!

**Answer/Grading:** If it were legal we could add different types of objects to strL (for example, String and Integer objects.)