EE 422C – Midterm Exam – Summer 2016

|  |  |  |
| --- | --- | --- |
| Problem  Number | Question | Points  Possible |
| 1 | Short Answers | 36 |
| 2 | Recursion | 24 |
| 3 | Scanner | 15 |
| 4 | Polymorphism | 26 |
| 5 | Generics | 17 |
| SCORE OUT OF: 118 | | |

***Problem 1 – Short Answers (36 points)***

For all parts of this problem, if the code does not compile, write CE. If there is a runtime error, write RE. If there is an infinite loop, write IL.

Questions A and B use the following method.

A. [2, 6] 2 points, 1 for each correct number. No points off for missing comma or square brackets.

B. CE 2 points

C. true (1 point)

D. 1.57 (1 point)

E. D (1 point)

F. 20, 20, 10, 10 (4 points)

1. .Legal, CE (2 points)
2. Legal, CE (2 points)

I. false (2 points)

J. false (2 points)

K. false desk (2 points)

L. CE (2 points)

M. 100 s: 100 (3 points) Partial credit

N. 100 true (2 points)

O. 130 (2 points)

P. CE (2 points)

Q. true (2 points)

R. RE (2 points)

***Problem 2 – Recursion (24 points).***

(a) 8 points

1 points for finding null base case, 1 point for returning false

1 points for finding equals base case, 1 point for returning true

4 points for recursive case

public boolean find(LLNode<T> start, T value) {

if (start == null)

return false;

else if (start.data.equals(value))

return true;

else

return find(start.next, value);

}

(b) (4 points)

1 points for each of the first 4 lines below.

(c) (12 points) for the remaining 5 lines. -1 for no null check of child in one or both lines before accessing child. -1 if last return is missing (i.e. if node is a leaf).

public boolean find(BSTNode root, Integer value) {

if (root == null)

return false;

if (root.data.equals(value))

return true;

if (root.left != null && root.left.data > value)

return find(root.left, value);

if (root.right != null && root.right.data < value)

return find(root.right, value);

return false;

}

***Problem 3 – Scanner (15 points):***

3 points declaring or initializing some sort of container.

2 points for loop with hasNext

2 point for calling s.next

2 points for adding String

3 points for avoiding duplicates by using a HashSet or other means.

2 point for loop to go through stored container

1 point for printing out each word.

public static void printNonDuplicates(Scanner s) {

HashSet<String> items = new HashSet<String>();

while (s.hasNext()) {

items.add(s.next());

}

for (String word: items)

System.out.println(word);

}

***Problem 4 – Polymorphism (26 points)***

In the following questions, the classes Rectangle, Triangle, and Circle all extend Shape, and Square extends Rectangle.

(a) (12 points)

1 pt return value, 1 point for ?, 1 point for Collection of Shape.

1 points for ArrayList, 1 point for ArrayList of Rectangles. If they put a extends or super, it is not correct (-1). ArrayList of Shapes -1.

2 points for for loop.

1 point for instanceof check.

1 point for add, 1 point for cast

1 point for return, 1 point for returning correct value.

public ArrayList<Rectangle> method (Collection<? extends Shape> shapes) {

ArrayList<Rectangle> recList = new ArrayList<Rectangle>();

for (Shape shape : shapes) {

if (shape instanceof Rectangle) {

if (shape instanceof Rectangle)

recList.add((Rectangle) shape);

}

}

return recList;

}

(b) (2 points)

2 points if they circle 3. 0 if they circle anything else, even if it includes 3.

(c) (2 points)

abstract class

Some code like getColor can be shared, so abstract class is better than interface.

(d) (2 points) (if their syntax for the keyword abstract is not quite right, it's OK.

public abstract Shape clone();

(e) (8 points)

1 point for lines 1-3. 2 points for line 4 and 5, including the cloning of the point and Color. 1 point for return. Returning anything other than a Circle (such as an Object) is not OK (-1), as the description says that an object of the same type is returned.

public Circle clone () {

Circle c = new Circle();

c.radius = this.radius;

c.center = this.center.clone();

c.color = this.color.clone();

return c;

}

***Problem 5 – Generics (17 points)***

2 points for header, with generic. -1 for no generic

3 point for going through values with a loop

2 point for detecting and avoiding nulls

3 point for finding pos

3 point for returning ele.value

4 points for returning default

public E get(int pos) {

for (ListElem<E> ele : values) {

if (ele != null && ele.position == pos)

return ele.value;

}

return defaultValue;

}

}