Name:____ (PRINT CLEARLY)

Lab Section:________ Grader:____________________________________

Quiz 5A – December 6

CS 2102 B19

1. Review the following HashMap code.

```
import java.util.HashMap;

public class Dates {

     private HashMap<Integer, String> dates;

     public Dates() {

          this.dates = new HashMap<Integer, String>();
     }

     public String getDate(int thisDate) {

          return this.dates.get(thisDate);
     }

     public Dates addDate(int thisDate, String dateName) {




          String oldDate = this.dates.put(thisDate, dateName);

          return this;
     }
}
```

(4 points) Modify the above code to handle collisions.

```
import java.util.HashMap;

public class Dates {

	private HashMap<Integer, LinkedList<String>> dates;

	public Dates() {
		this.dates = new HashMap<Integer, LinkedList<String>>();
	}

	public LinkedList<String> getDate(int thisDate) {

		return dates.get(thisDate);
	}


	public Dates addDate(int thisDate, String dateName) {
		LinkedList<String> curDates = this.getDate(thisDate);
		curDates.add(dateName);
		String oldDate = dates.put(thisDate, curDates);
		return this;
	}
}
```

Scoring:

+1 Replaces all instances of `String` with `LinkedList<String>` EXCEPT in the `addDate()` parameter

- - 0.5 if `addDate()` parameter is also modified OR it's unclear that this parameter should not be modified
- - 0.5 for each unmodified `String` that should be modified to a `LinkedList` OR it's unclear that certain `String`s should be modified (up to 1 point)

+3 in `addDate()`, program gets the existing LinkedList (+1), adds the new item to it(+1), then puts the new LinkedList back in the HashMap (+1)

- Students who are writing code can use either `getDate()` or the HashMap `get()` function.
- Okay (but not required) if return type is changed to void, but then the return statement should also be removed (- 0.5 for forgetting this)
- No penalty if the student changes the return type to a LinkedList<String>, even if they don't change the type of the `oldDate` variable

2. (3 points) Review the following code. What will be printed to the console when we run the program? Explain why. Be sure to include a discussion of exceptions in your explanation.

```
public class Main {
      public static void main(String[] args) {

          private LinkedList<String> heroes = new LinkedList<String>();

           heroes.addLast("Wonder Woman");
           heroes.addLast("Iron Man");
           heroes.addLast("Captain Marvel");

           try {
                 System.out.println(heroes.get(3));
           }
           catch (IndexOutOfBoundsException e) {
                 System.out.println("Exception caught!");
           }

      }
}
```

“Exception caught”. We have an array with only three items, so they will have index values 1 – 2. When we try to access an item at index 3, that item does not exist, and this will generate an `IndexOutOfBoundsException` Since we are making the call inside a try/catch block, this block will catch the exception and print out the corresponding message.

Scoring:

+1 Correct string printed to console.

+2 Correct explanation. (-1 if explanation does not include a discussion of exceptions, including the exception generated and how the try/catch block is used.)

3. (3 points) Briefly explain how the `hashCode()` method is used when getting values from or putting values in a hash map.

The `hashCode()` method takes a hash map key and transforms it into a number that tells the program exactly where in the hash map to store the corresponding value.

Scoring:

+3 Explanation indicates an understanding that `hashCode()` turns a key (+1) into an integer (+1) that serves as a hash map address (+1).