Points off	1	2	3	4	5		Total off	Net Score

CS 314 – Midterm 2 – Spring 2013

Your Name_____

Your UTEID _____

Circle yours TA's name: Donghyuk Lixun Padmini Zihao

Instructions:

1. There are **5** questions on this test. The test is worth 70 points. Scores will be scaled to 175 for grade center.
2. You have 2 hours to complete the test.
3. You may not use a calculator or any other electronic devices while taking the test.
4. When writing a method, assume the preconditions of the method are met.
5. When writing a method you may add helper methods if you wish.
6. When answering coding questions, ensure you follow the restrictions of the question.
7. Test proctors will not answer any questions.
8. When you complete the test show the proctor your UTID, give them the test and any scratch paper, and please leave the room quietly.

1. (1 point each, 20 points total) Short answer. Place your answers on the attached answer sheet.
   a. If a question contains a syntax error or other compile error, answer "Compile error".
   b. If a question would result in a runtime error or exception answer "Runtime error".
   c. If a question results in an infinite loop answer "Infinite loop".
   d. Recall when asked for Big O your answer shall be the most restrictive correct Big O function. For example Selection Sort has an average case Big O of $O(N^2)$, but per the formal definition of Big O it is correct to say Selection Sort also has a Big O of $O(N^3)$ or $O(N^4)$.

A. What is returned by the method call `a(27, 3)`?

```
public int a(int n, int c) {
     if(n < c)
          return 3;
     else
          return 2 + a(n - c, c + 3);
}
```

B. What is returned by the method call `b("012345678901234")`?

```
public int b(String st) {
     if(st.length() <= 3)
          return st.length();
     else
          return st.length() + b(st.substring(4));
}
```

C. What is the order (Big O) of method `c`? N = `n`.

```
public int c(int n) {
     if(n <= 0)
          return 1;
     else
          return 1 + c(n / 4);
}
```

D. What is returned by the method call `d(10)`?

```
public int d(int n) {
     if(n <= 2)
          return 3;
     else
          return n % 3 + d(n - 2) + d(n - 4);
}
```

E. What is output by the following code if `list1` is an `ArrayList<Integer>` that contains the following values: `[-2, 5, 2, -2, 1, 3]`

```
Iterator<Integer> it1 = list1.iterator();
for(int i = 0; i < 2; i++) {
     System.out.print(it1.next() + " ");
     it1.remove();
     System.out.print(it1.next() + " ");
}
```

F. What is output by the following code?

```
Map<String, Integer> map1 = new Map<String, Integer>();
map1.put("A", 2);
System.out.print(map1.size());
```

G. What is output by the following code if `list2` is an `ArrayList<Integer>` that contains the following values: `[2, 3, 2, 3, 4, 6]`

```
Iterator<Integer> it2 = list2.iterator();
while(it2.hasNext()) {
     if(it2.next() % 2 == 0)
          System.out.print(it2.next());
}
```

H. What is the worst case order (Big O) of method `h`? N = `data.length`

```
public List<Integer> h(int[] data, int min) {
     List<Integer> result = new LinkedList<Integer>();
     for(int x : data)
          if(x > min)
               result.add(0, x); // 0 is index  at which to insert
     return result;
}
```

I. What is the worst case order (Big O) of method `h` if the line

```
List<Integer> result = new LinkedList<Integer>();
```

is changed to

```
List<Integer> result = new ArrayList<Integer>(data.length);
// create an ArrayList with initial capacity of data.length
```

J. Given an array of 1,000,000 `int`s in ascending order it takes 10 seconds to complete 100,000 searches in the array, using the binary search algorithm. What is the expected time to complete 200,000 binary searches given an array of 2,000,000 `int`s in ascending order?

K. A method uses the insertion sort algorithm to sort an array of `double`s. It takes the method 2 seconds to sort an array with 50,000 distinct `double`s (no repeats) that are initially in random order. What is the expected time for the method to sort an array with 200,000 distinct `double`s that are initially in random order?

L. What is the result of the following postfix expression? (single integer for answer)

```
3  6  2  -  *  10 +
```

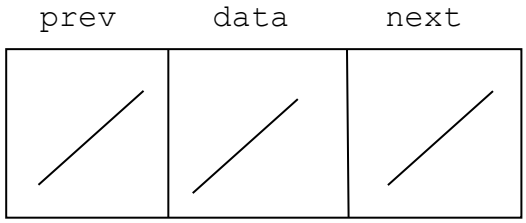
M. What is output by the following code? Assume the `Stack` class is the one developed in lecture.

```
int[] sData = {3, 5, 1, -1, 2, 1};
Stack<Integer> st1 = new Stack<Integer>();
for(int x : sData) {
     st1.push(x);
     st1.push(x);
}
for(int i = 0; i < 5; i++)
     System.out.print(st1.pop() + " ");
```

N. What is output by the following code? Assume the `Stack` class is the one developed in lecture, with the addition of a `size` method that returns the number of elements currently in the `Stack`.

```
Stack<Integer> tricky = new Stack<Integer>();
for(int i = 0; i < 12; i += 2)
     tricky.push(i + 1);
for(int i = 0; i < tricky.size(); i++)
     System.out.print(tricky.pop() + " ");
```

O. Draw the variables, references, and objects that exist after the following code executes. Draw node objects as shown below and boxes for variables. The example has all instance variables set to `null`. The example does not show any of the variables that actually refer to the node object. You must show all variables and their references in your drawing. Use arrows to show references and a forward slash to indicate variables that store `null`. Assume the node class is the doubly linked node from the linked list assignment.




```
DoubleListNode<Object> n1
     = new DoubleListNode <Object>(null, "AB", null);
                  // parameters are prev, data, next
n1.setPrev(n1);
DoubleListNode<Object> n2;
n2 = new DoubleListNode <Object>(n1 , n1, null);
n1.setNext(n2);
n2.setNext(n1.getPrev().getNext());
```

P. You have an array with 1,000 distinct elements in random order.

You have to search the array 10,000 times to determine if a given element is present or not.

What will result in less work? Sorting the array with quicksort and then doing the searches using binary search OR just doing the searches with linear search. (without sorting)

You must justify your answer with calculations.

Q. Consider the following resize method for an array based list:

```
private void resize() {
     E[] temp = (E[]) new Object[size + 10];
     for(int i = 0; i < size; i++)
          temp[i] = container[i];
     container = temp;
}
```

It takes 5 seconds for the following code to run.
The `GenericList` class uses the `resize` method shown above.

```
GenericList<Integer> listQ = new GenericList<Integer>();
for(int i = 0; i < 10000; i++)
     listQ.add(i);
```

What is the expected time for the code to run if the `boolean` condition is changed to `i < 30000`?

R. This question involved the same code from question 1.Q. Assume the line

```
E[] temp = (E[]) new Object[size + 10];
```

in the `resize` method is changed to

```
E[] temp = (E[]) new Object[size * 2 + 1];
```

It takes 0.5 seconds for the code segment to run when the `boolean` condition is `i < 10000`. What is the expected time for the code to run if the `boolean` condition is changed to `i < 30000`?

S. What is the order (Big O) of method `total1`? N = `list.size()`.

```
public double total1(LinkedList<Double> list) {
     double total = 0.0;
     for(int i = 0; i < list.size(); i++)
          total += list.get(i);
     return total;
}
```

T. What is the order (Big O) of method `total2`? N = `list.size()`.

```
public double total2(LinkedList<Double> list) {
     double total = 0.0;
     for(double d : list)
          total += d;
     return total;
}
```

2. Maps - 13 points. Complete an instance method for the `Names` class from assignment 4 that counts the number of `NameRecord` objects that are **not** ranked in the last N decades. N is a parameter to the method.

For this question the `Names` class stores its `NameRecord`s in a `Map` named `data`. The keys of the `Map` are `String`s (the name such as "Olivia", "Bob", "Abe", and "Buffy") and the values are the associated `NameRecord` objects.

The `countUnrankedNames` method accepts a single `int, n,` as a parameter.

All `NameRecord` objects stored in the instance variable named `data` have the same number of ranks. Their `numDecade` methods all return a value equal to the `NUM_DECADES` class constant in `Names`.

Consider this **example**: Assuming `n = 6` and `NUM_DECADES = 11`

key (name)	ranks in associated `NameRecord` object
`"Abe"`	`248 328 532 764 733` **`0 0 0 0 0 0`**
`"Buffy"`	`0 0 0 0 0` **`0 0 678 0 0 0`**

The method would return 1 if those were the only two `NameRecord` objects. The `NameRecord` associated with `Abe` is not ranked in the last 6 decades. Buffy does not count because it is ranked in at least one of the last 6 decades. If `n` equaled 1, 2, or 3 the method would return 2. If `n` was greater than 6 then the method would return 0. (Assuming those were the only two `NameRecord` objects in the map.)

Note, `NameRecord` objects do not store zeros for unranked decades. They store the value `UNRANKED`.

The `NameRecord` class for the question:

```
public class NameRecord {

     // indicates this Name Record is not ranked in a given decade
     public static final int UNRANKED = 10311225;

     /* Return the rank of this NameRecord for the given decade.
        0 <= decade < numDecades()
        The returned value is between 1 and 1000 inclusive or
        equal to UNRANKED. */
     public int getRank(int decade)

     /* return the number of decades this NameRecord is ranked
     including unranked decades.*/
     public int numDecades()
```

Recall these methods from the `Map` interface:

- `Set<K>` **`keySet`**`()` - Returns a Set view of the keys contained in this map.
- `V` **`get`**`(Object key)` - Returns the value to which the specified key is mapped, or null if this map contains no mapping for the key.
- `V` **`put`**`(K key, V value)` - Associates the specified value with the specified key in this map
- `V` **`remove`**`(K key)` - Removes the mapping for a key from this map if it is present.

Recall this method from the `Set` interface:

- `Iterator<E>` **`iterator`**`()` Returns an iterator over the elements in this set.

Recall these methods from the `Iterator` interface:

- `boolean` **`hasNext`**`()` – Returns true if the iteration has more elements.
- `E` **`next`**`()` - Returns the next element in the iteration.
- `void` **`remove`**`()` – Removes from the underlying collection the last element returned by the iterator

And finally the `Names` class itself:

```
public class Names {

     private Map<String, NameRecord> data;
     private int NUM_DECADES; // number of decades for each NameRecord

     // pre: 0 < n < NUM_DECADES
     // post: as described in the question. this object is unaltered
     public int countUnrankedNames(int n) {
```

You are not allowed to use any other methods or classes except those listed in this question.

Your methods shall be as efficient as possible given the constraints of the question.

Complete the method on the next page.

```
// pre: 0 < n < NUM_DECADES
// post: as described in the question. this object is unaltered
public int countUnrankedNames(int n) {
```

3. Linked Lists - 12 points. Complete the `insertAfter` instance method for a `LinkedList` class. The method inserts a given element after each occurrence of a target element in the list.
The method returns the number of elements added.

- You may not use any other methods in the `LinkedList` class unless you implement them yourself as a part of your solution.
- **Do not use recursion in your solution.**
- The `LinkedList` class uses singly linked nodes.
- The list has a reference to the first node in the list. There is no last or size instance variables.
- When the list is empty, `first` is set to `null`.
- None of the data in the list equals `null`.
- If the list is not empty the last node in the list has its next reference set to `null`.
- You may use the `Node` class and the `Object equals` method.
- **You may not use any other Java classes or native arrays.**
- **Your method shall be as efficient as possible given the constraints of the question.**

```
public class LinkedList<E> {

     private Node<E> first; // first node in chain of nodes

}
```

The `Node` class.

```
public class Node<E> {
     public Node(E item, Node<E> next)
     public E getData()
     public Node<E> getNext(
     public void setData(E item)
     public void setNext(Node<E> next)
}
```

Examples. The first parameter is the target and the second parameter is the insert value.

```
[].insertAfter(A, B) ->  resulting list [], returns 0
[A].insertAfter(A, B)  ->  resulting list [A, B], returns 1

[A, A, A].insertAfter(A, B)
     ->  resulting list[A, B, A, B, A, B], reuturns 3

[A, C, C, A].insertAfter(A, B)
     ->  resulting list [A, B, C, C, A, B], returns 2

[C, C, C].insertAfter(A, B)  ->  resulting list [C, C, C], returns 0
[B, B, B].insertAfter(A, B)  ->  resulting list [B, B, B], returns 0
```

Complete the following method:

```
// pre: tgt != null, insertVal != null, !tgt.equals(insertVal)
// post: as described in the question, returns the number
// of elements added to this LinkedList
public int insertAfter(E tgt, E insertVal) {
```

4. Working with data structures - 13 points. Write a method that accepts a queue and a target object and then determines the average distance of the occurrences of the target object from the front of the queue.

Consider the following example:

```
front                          back
   [A, B, A, B, C, C, A, A, C]
    0  1  2  3  4  5  6  7  8
```

If the target element is A, it occurs at positions 0, 2, 6, and 7. The average distance of the occurrences of A from the front of the queue is `(0 + 2 + 6 + 7) = 15 / 4 = 3.75`.

If the target element is not present in the queue, the method shall return `-1`.

The only methods you may use are the queue methods `enqueue, dequeue, isEmpty,` and `front`.

The only class you may use is the `Queue` class (including a temporary queue) and the `Object equals` method.

This method is NOT an instance method in the Queue class. It is a client method.

Your method shall be as efficient as possible given the constraints.

The queue parameter must be restored to its original state when the method is finished.

```
// pre: q != null, tgt != null
// post: per the question description
public double getAveDistanceFromFront(Queue<Object> q, Object target) {
```

Complete the method on the next page.

```
// pre: q != null, tgt != null
// post: per the question description
public double getAveDistanceFromFront(Queue<Object> q, Object target)
{
```

5. Recursion - 12 points. In the book *Thinking Recursively with Java* by Eric Roberts the author mentions the card game cribbage. "One aspect of determining the score comes from computing the number of distinct card combinations whose values add up to 15 with aces counting as 1 and all face cards (jack, queens, and kings) counting as 10."

For this questions aces will count as 11 not 1.

We will represent card values with `int`s between 2 and 11 inclusive. If we have these five values: (11, 5, 10, 4, 9) there are two distinct combinations that add to 15.

11 + 4 5 + 10

If we have these five values: (5_1, 5_2, 5_3, 5_4, 10) there are eight different combinations that add to 15. (subscripts added for clarity)

$5_1 + 5_2 + 5_3$ $5_1 + 5_2 + 5_4$ $5_1 + 5_3 + 5_4$ $5_2 + 5_3 + 5_4$
$5_1 + 10$ $5_2 + 10$ $5_3 + 10$ $5_4 + 10$

For this question alter the rules so that each card can either count as its value, double it value, or triple its value. For example, given the values (5_1, 5_2, 10) there are six combinations that equal 15.

5_1 * 3 (triple value of 5_1) 5_2 * 3 (triple value of 5_2) 5_1 * 2 + 5_2 (double value of 5_1)
5_2 * 2 + 5_1 (double value of 5_2) $5_1 + 10$ $5_2 + 10$

Note, a card cannot be used more than once in a given combination.
It is **not valid** to create a combination equal to 15 with 5_1 doubled plus 5_1 itself.

For each card your choices when forming a combination are to not use the card, to use its value, to use double its value, OR to use triple its value.

NOTE: The number of values will not always equal 5. It could be more or less. The target value is not always 15. It is a parameter to the method.

Write a recursive backtracking method to determine the number of combinations of values that equal the target value given the rules above.

You may use native arrays, but no other Java classes or methods.

Complete the helper method for the following method.

```
// pre values != null, all values between [2, 11]
public int cardCombos(int[] values, int tgt) {
     return helper(values, 0, tgt);
}
```

COMPLETE THE HELPER METHOD ON THE NEXT PAGE. FOR THIS QUESTION YOU MAY NOT ADD ANY OTHER HELPERS OR CHANGE THE PARAMETERS.

```
private int helper(int[] values, int valueIndex, int tgt) {
```

Question 1 Answer Sheet.

Name_____ UTEID _____

Bonus question: _____

A. _____

B. _____

C. _____

D. _____

E. _____

F. _____

G. _____

H. _____

I. _____

J. _____

K. _____

L. _____

M. _____

N. _____

O. _____

P. _____

Q. _____

R. _____

S. _____

T. _____