Exam Number:

Points off	1	2	3	4	5	Total off	Net Score

CS 307 – Midterm 2 – Spring 2011

Name_____

UTEID login name _____

TA's Name: Dan Muhibur Oliver (Circle One)

Instructions:

1. Please turn off your cell phones and other electronic devices.
2. There are 5 questions on this test.
3. You have 2 hours to complete the test.
4. You may not use a calculator on the test.
5. When code is required, write Java code.
6. When writing methods, assume the preconditions of the method are met. Do not write code to check the preconditions.
7. On coding question you may add helper methods if you wish.
8. After completing the test please turn it in to one of the test proctors and show them your UTID.

1. (2 points each, 30 points total) Short answer. Place you answers on the attached answer sheet.

- If the code contains a syntax error or other compile error, answer "Compile error".
- If the code would result in a runtime error / exception answer "Runtime error".
- If the code results in an infinite loop answer "Infinite loop".

Recall that when asked for Big O your answer should 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 $O(N^4)$. I want the most restrictive correct Big O function. (Closest without going under.)

A. What is returned by the method call `a(4)`?

```
public static int a(int x) {
      if(x <= 0)
            return 0;
      else
            return x * x + a(x - 1);
}
```

B. What is returned by the method call `b(13)`?

```
public static int b(int x) {
      if(x >= 20)
           return x / 2;
      else
            return x / 3 + b(x + 3);
}
```

C. What is printed when the method call `c1()` is made?

```
public static void c1() {
      int[] data = {5, 10, 15, -10, 10, -5, 5, 10, 10, 15, 20, 15};
      c2(2, data);
}

public static void c2(int i, int[] vals) {
      if(i >= vals.length)
            System.out.print("A");
      else if(vals[i] > 5) {
            System.out.print("B");
            c2(i + 2, vals);
            System.out.print("C");
      }
      else {
            System.out.print("D");
            c2(i + 1, vals);
            System.out.print("E");
      }
}
```

D. What is returned by the method call `d(8)`?

```
public static int d(int x){
      if(x <= 1)
            return x;
      else
            return d(x - 2) + d(x - 1);
}
```

E. What is the order (Big O) of method `e`? N = `data.length`

```
public static int e(int[] data) {
      int accum = 0;
      for(int i = 0; i < data.length; i++)
            for(int j = data.length - 1; j >= i; j--)
                  accum += data[i] * data[j];
      return accum;
}
```

F. What is the worst case order (Big O) of method `f`? N = `list.length`.

```
// pre: list != null, list.length > 4
public int f(int[] list){
     int total = 0;
     int j = list.length - 1;
     for(int i = 0; i < 4; i++) {
               if(list[i] == list[j] || list[i] % 10 == list[j] %
     10)
               total++;
          j--;
     }
     return total;
}
```

G. What is the T(N) for method g? Recall, T(N) is the function that represents the *actual* number of executable statements for a function or algorithm. N = `values.length`.

```
public static double g(double[] values){
     double total = 0;
     for(int i = 0; i < values.length; i++)
          total += values[i];
     for(int i = 0; i < values.length * 3; i++) {
          double temp = values[i / 3] * i;
          total = total + temp;
     }
     return total;
}
```

H. What is the best case order (Big O) of method `h`? N = `data.size()`. Assume the equals method for `Integer` is O(1).

```
public static int h(LinkedList<Integer> data, char ch) {
     int total = 0;
     for(int i = 0; i < data.size(); i++)
          for(int j = i + 1; j < data.size(); j++)
               for(int k = j + 1; k < data.size(); k++)
                    if(data.get(i).equals(data.get(j)) &&
                              data.get(j).equals(data.get(k)))
                         total++;
     return total;
}
```

I. What is the order (Big O) of method `i2` ? N = `org.length`. The `ArrayList` constructor is O(1).

```
public static ArrayList<Double> i2(double[] org){
    ArrayList<Double> result = new ArrayList<Double>();
    for(int i = 0; i < org.length; i++)
        result.add(result.size() / 2, org[i]);

     /* result.size() / 2 is the position org[i] is added to the
          list */

    return result;
}
```

J. An algorithm is O(N). It takes 0.25 seconds for the method to run when N = 1,000,000. What is the expected time for the method to run when N = 10,000,000?

K. A sorting method uses the insertion sort algorithm. It takes 10 seconds for the method to sort 100,000 distinct integers in random order. What is the expected time for the method to sort 400,000 distinct integers in random order?

L. You do not have the source code to analyze a method so you have run a series timing experiments on it. Based on the following results what is the most likely order (Big O) for the method?

N	Time to complete
1000	1 second
2000	7.5 seconds
4000	64.6 seconds
8000	510 seconds

M. A method is $O(2^N)$. When N = 100 the method takes 10 seconds to run. What is the expected time for the method to run when N = 110?

Consider the following `Node` class for questions N and O.

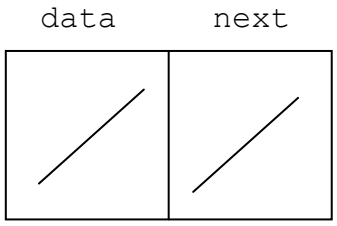
```
public class Node {
     public Object data;
     public Node next;

     public Node(Object d, Node n) {
          data = d;
          next = n;
     }
}
```

N. What is output by the following code?

```
Node n1 = new Node(12, new Node(-7, new Node(6, null)));
Node n2 = n1.next.next;
n2.next = n1.next;
Node n3 = new Node(13, n2.next.next);
System.out.print(n3.next.next.data);
```

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 both 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`.




```
Node n4 = new Node("Z", new Node("X", null));
Node n5 = new Node(null, n4.next);
n4.next.next = n5;
Node n6 = n4.next;
```

2. (Implementing Data Structures, array based lists, 15 points) Write a method for the `GenericList` class we developed in lecture that removes the first N elements of the list.

- The `GenericList` is generic based on Java's inheritance requirement and polymorphism, not the Java generic syntax.
- `GenericList`'s internal storage is a native array of `Objects`. There may be extra capacity in the array. **Do not create a new array for the `GenericList` in this question**
- The array variable named `container` will never equal `null`.
- The elements of the list use 0 based indexing. The position of an element in the list is equal to its index in the array.
- You may not use any other methods from the `GenericList` class unless you implement them yourself in this question.
- Your method must be O(1) ***space***, meaning you cannot use temporary arrays, lists, or other data structures.
- All elements of `container` that do not refer to elements of the list **must be set to `null`.**

Complete the following method in the `GenericList` class:

```
/*   pre: 0 <= num <= size()
     post: the first num elements are removed from this list. The
     remaining elements are shifted to the left.
     size = old size - num. */
public void removeFrontPortion(int num)
```

Examples of results of `removeFrontPortion`:

```
[A, B, A, C].removeFrontPortion(0) -> [A, B, A, C]
[A, B, A, C].removeFrontPortion(1) -> [B, A, C]
[A, B, A, C].removeFrontPortion(2) -> [A, C]
[A, B, A, C].removeFrontPortion(3) -> [C]
[A, B, A, C].removeFrontPortion(4) -> []
[].removeFrontPortion(0) -> []
```

```
public class GenericList{

     private Object[] container;
     private int size;
```

Implement the method on the next page:

```
/*   pre: 0 <= num <= size()
     post: the first num elements are removed from this list. The
     remaining elements are shifted to the left.
     size = old size - num. */
public void removeFrontPortion(int num)
```

3. (Sets, 15 points) Write a method for the `AbstractSet` class from assignment 8 that returns the size of the difference between `this` set and another set sent as a parameter. This method does not alter either set or create a new set. Recall the difference of `this` set and another set is all of the elements in `this` set that are not in the other set.

- The only method you can use from `ISet` is the `iterator` method. If you want to use any other methods from `ISet` you must implement them yourself.
- You may not use the foreach loop. Assume `ISet` **does not** implement the `Iterable` interface.
- Recall, the `AbstractSet` class does not have any instance variables.
- Do not make any explicit references to `UnsortedSet` or `SortedSet`.
- You can use all of the methods from the `Iterator` interface.
- Your method must be O(1) ***space***, meaning you cannot use temporary arrays, lists, or other data structures besides `iterator` objects which are O(1) space.
- Neither set is altered as a result of this method call.

Examples:

```
(2, 1, 6, 4).sizeOfDifference( (4, 3, 2, 6) ) -> 1
(1, 2). sizeOfDifference ( (2, 1) ) -> 0
(1, 5, 2, 3). sizeOfDifference ( () ) -> 4
().sizeOfDifference ( () ) -> 0
(4, 1, 2, 15). sizeOfDifference ( (12, 5, -1, 7) ) -> 4
```

`Iterator<E>` **`iterator`**`()`

Returns an iterator over the elements in this set.

Recall the 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.

Complete the following method on the next page. Recall, the method is part of the `AbstractSet` class.

```
// pre: other != null
// post: return the size of the difference of this set and other.
// this set and other are not altered as a result of this method
// call
public int sizeOfDifference(ISet<E> other) {
```

```
public abstract class AbstractSet<E> implements ISet<E> {

// pre: other != null
// post: return the size of the difference of this set and other.
// this set and other are not altered as a result of this method
// call
public int sizeOfDifference(ISet<E> other) {
     assert other != null;
```

4. (Linked Lists 20 points). Write a method an `equals` method for the `LinkedList` class. This method overrides the equals method from the `Object` class.

The properties of the LinkedList class:

- The list uses singly linked nodes that store one piece of data and a reference to the next node in the list.
- The only instance variables in the `LinkedList` class are a reference to the first node in the linked list and the size of the list.
- If the list is empty the reference to the first node is set to `null`.
- The last node's next reference is set to `null`.
- The `LinkedList` and `Node` objects are generic based on Java's inheritance requirement and polymorphism, not the Java generic syntax.

Your method must be O(1) ***space***, meaning you cannot use temporary arrays, lists, or other data structures whose size depends on the number of elements in the linked list.

You may not use any other classes or methods, except the Node class and the equals method from any class.

Your method should be as efficient as possible given the other constraints.

Recall that since this method is in the `LinkedList` class, you have access to all `LinkedLists'` instance variables.

Here are some examples and the expected results for various lists.

```
[].equals(null) -> false
[1, 2, 1].equals(null) -> false
[1, 2, 1].equals([A, B, C]) -> false
[1, 2, 1].equals([1, 2]) -> false
[1, 2, 1].equals([2, 1, 1]) -> false
[1, 2, 1].equals([1, 2, 1]) -> true
[1, 2, 1].equals([1, 2, 1, 2]) -> false
```

Here is the Node class:

```
public class Node {
     public Node(Object data, Node next)

     public Object getData()
     public Node getNext()

     public void setData(Object data)
     public void setNext(Node next)
}
```

```
public class LinkedList {

     private Node first;
     private int size;

// pre: none
// post: return true if other is a LinkedList and has the same
// elements as this LinkedList, in the same order. Otherwise
// return false. Neither this or other are altered as a result
// of this method call.
public boolean equals(Object other) {
```

5. (Recursion, 20 points) Complete a method that determines if a sailboat with an initial location, can reach a destination. The wind is blowing to the east, so the sailboat may only move to the north, east, or south. It may not move west. On the examples shown below, north is up, east is right, south is down, and west is left.

The sailboat moves 1 cell at a time. For example, the first move may be to the east moving the sailboat one column to the right. The next move could be to the north, moving the sailboat one row up.

The sea the sailboat is on is depicted as a 2D array of chars. Water is depicted with a `'0'` and rocks are depicted with a `'%'`. The sailboat may not enter locations with a `'%'`. Consider the example sea.

```
0000000000
0000%%%000
00%000%000
00%%%0%000
000000%000
%%%%%%%%%%
```

If the sailboat starts at cell 4, 4 and its goal is 0, 8, there is no way for the boat to move to that destination. Recall, the boat may not move to the west, (left in the 2D array). In the following example, S indicates the initial location of the boat and * is the destination. Note, the 2D array for the method you are writing will initially only contain the characters `'0'` and `'%'`.

```
00000000*0
0000%%%000
00%000%000
00%%%0%000
0000S0%000
%%%%%%%%%%
```

Complete the following method. You may not use any other Java classes or methods. You may alter the 2D array of chars in any way you want.

```
/*   pre: map != null, map is a rectangular matrix. Initially all
elements of map are '0' or '%', but this may change during
recursive calls. Initially startRow, startCol, destinationRow, and
destinationCol are all within the bounds of map and
map[startRow][startCol] == 'O'.
     post: return true if it is possible for a sailboat at
startRow, startCol to reach destinationRow, destinationCol through
a series of up, down, or right moves that do not hit any rocks,
the '%' char.
*/
public boolean canSail(char[][] map, int startRow, int startCol,
                       int destinationRow, int destinationCol) {
```

```
public boolean canSail(char[][] map, int startRow, int startCol,
                       int destinationRow, int destinationCol) {
```

No test material this page

Name:_____

TA's Name: Dan Muhibur Oliver (Circle One)
Answer sheet for question 1, short answer questions

A. _____

B. _____

C. _____

D. _____

E. _____

F. _____

G. _____

H. _____

I. _____

J. _____

K. _____

L. _____

M. _____

N. _____

O. place answer for O on next page.

Answer for 1.O: