Exam Number:

Points off	1	2	3	4	5	Total off	Net Score

CS 307 – Midterm 2 – Fall 2008

Name_____

UTEID login name _____

TA's Name: Mikie Ron Sarah (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. In 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 the worst case Big O of the following method? N = `n`

```
public int a(int n){
      int count = 0;
      int limit = 3 * n;
      for(int i = 1; i <= limit; i++){
            if(  limit % i == 0 )
                  count++;
      }
      return count;
}
```

B. What is the best case Big O of the following method? N = `data.length`

```
public int b(int[] data, int tgt){
      int count = 0;
      for(int i = 0; i < data.length; i++){
            for(int j = 0; j < data.length; j++){
                  if( data[i] * data[j] == tgt )
                        count++;
            }
      }

      for(int i = 0; i < data.length; i++){
            if( data[i] == tgt )
                  count++;
      }

      return count;
}
```

C. What is the Big O of the following method? N = `data.size()`

```
public int c(ArrayList<Integer> data){
      int sum = 0;
      for(int i = 1; i < data.size(); i *= 3)
            sum += data.get(i);
      return sum;
}
```

D. What is the Big O of the following method? N = `data.length`. Method `numPresent` is O(N).

```
public int d(int[] data){
      int sum = 0;
      for(int i = 1; i <= data.length; i *= 2)
            sum += numPresent(data, i);
      return sum;
}
```

E. What is the Big O of adding an element at position 0 to an array based list that already contains N elements?

F. What is the Big O of adding an element at position 0 to a linked list that already contains N elements?

G. What is the Big O of the following method? N = `data.length`.

```
public int g(LinkedList<Integer> data){
      int total = 0;
      for(int i = 0; i < data.size(); i++)
            for(int j = i; j < data.size(); j++)
                  total += data.get(i) * data.get(j);
      return total;
}
```

H. What is the T(N) of the following method? N = `data.length`. `data` is a square matrix with the same number of columns as rows. `other` will be the same size as `data`.

```
public int sub(int[][] data, int[][] other){
      int total = 0;
      int temp = 0;
      for(int r = 0; r < data.length; r++){
            for(int c = 0; c < data.length; c++){
                  temp = data[r][c] - other[r][c];
                  total += temp * 2;
            }
      }
      return total;
}
```

I. Given an unsorted array of 10,000 integers, how many of the integers will be checked when doing a search for a value that is not present in the array?

J. A method is $O(N^2)$. It takes 1 second for the method to complete on a data set with 5,000 items. What is the expected time for the method to complete with a data set of 20,000 items?

K. A method is $O(\log_2 N)$. It takes 0.020 seconds for the method to complete on a data set with 1,000,000 items. What is the expected time for the method to complete with a data set of 2,000,000 items? Recall $\log_2 1{,}000{,}000 \sim= 20$.

L. Consider the following timing data for a sorting method. The method sorts data into ascending order. In each case there are N distinct integers (no duplicates) in the array and they are already sorted into ascending order. Of the sorting algorithms we studied which one is most likely used in the method?

N	time for sorting method to complete
200,000	1.000 second
400,000	2.000 seconds
800,000	4.000 seconds
1,600,000	8.000 seconds
3,200,000	16.000 seconds

M. What is returned by the method call `m(7)`?

```
public int m(int val){
      int result = 0;
      if( val <= 3 )
            result = 2;
      else
            result = m(val - 2) + (val - 2);
      return result;
}
```

N. What is returned by the method call `n("deep")`?

```
public String n(String s){
     if( s.length() == 0 )
          return "";
     else
          return s.charAt(0) + n( s.substring(1) )
               + s.charAt( s.length() - 1 );
}
```

O. Consider the following methods from the Java `ArrayList` class.

Method Summary	
void	**add**(E o) Appends the specified element to the end of this list.
void	**add**(int index, E element) Inserts the specified element at the specified position in this list.
E	**get**(int index) Returns the element at the specified position in this list.
E	**remove**(int index) Removes the element at the specified position in this list. Return the element that is removed from the list.
E	**set**(int index, E element) Replaces the element at the specified position in this list with the specified element. Returns the element that was previously at index.
int	**size**() Returns the number of elements in this list.

What is output when method `k` is called?

```
public void k(){
    ArrayList<String> ls = new ArrayList<String>();

    ls.add("A");
    ls.add("B");
    ls.add(0, "A");
    ls.set(1, "B");
    ls.add(0, ls.get( ls.size() - 1 ) );

    for(String s : ls)
        System.out.print( s );
}
```

Scratch Paper

2. (Implementing data structures, 20 points). Write an instance method for a `LinkedList` class named `numPresent`. The method returns the number of elements in the linked list that are equal to a given value.

This `LinkedList` class uses singly linked nodes. Each node stores a single `Object` and a link to the next node.

- This `LinkedList` class only contains a reference to the first node in the list.
- The last node in the list's `next` reference is set to `null`.
- When the list is empty `head` is set to `null`.
- The `LinkedList` does not contain a size instance variable.

Examples:
`[1, 5, 0, 5, 6].numPresent(1)` returns 1
`[1, 5, 0, 5, 6].numPresent(0)` returns 1
`[1, 5, 0, 5, 6].numPresent(5)` returns 2
`[1, 5, 0, 5, 6].numPresent(12)` returns 0
`[].numPresent(12)` returns 0
`[12].numPresent(12)` returns 1
`[12, 12, 12, 12].numPresent(12)` returns 4

- **You may not use any other methods in the `LinkedList` class, but you may create your own helper methods if you wish.**
- **Your solution must be O(1) space. In other words you cannot use an auxiliary array or ArrayList.**

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

      public Object getData()

      public Node getNext()

      public void setValue(Object value)

      public void setNext(Node next)
}

public class LinkedList
{     // points to the first node in the list
      // if the list is empty, head == null
      private Node head;

      // pre: value != null
      // post: return the number of elements in this list
      // equal to value
      public int numPresent(Object value){
      // complete this method on the next page
```

```
// pre: value != null
// post: return the number of elements in this list
// equal to value
public int numPresent(Object value){
```

3. (Implementing Data Structures 20 points) Consider an array based list class like the one we developed in class. The class is named `GenericList`. The class uses a native array of `Objects` as its internal storage container. The internal storage container may have extra capacity and thus its length will be greater than or equal to the length of the list that is being represented.

Complete an instance method that removes all elements from the list equal to some given value. The relative order of elements in the list that are not equal to the given value is unchanged.

- **Your method may not use any other methods in the `GenericList` class, but you may create your own helper methods if you wish.**
- **You may use native arrays in your solution.**
- **You may not use any other Java classes (other than native arrays) in your solution other than calling their `equals` methods.**

Here are some examples of the expected behavior of the method on various lists:

```
[].removeAll(A) -> []
[B, C, D, B, D, E].removeAll(A) -> [B, C, D, B, D, E]
[B, C, D, B, D, E].removeAll(B) -> [C, D, D, E]
[B, C, D, B, D, E].removeAll(D) -> [B, C, B, E]
[B, C, D, B, D, E].removeAll(122) -> [B, C, D, B, D, E]
[B, C, D, B, D, E].removeAll(b) -> [B, C, D, B, D, E]
```

```
public class GenericList{


     // container is never set to null
     private Object[] container; // internal storage container

     private int size; // number of elements in list


     // pre: value != null
     // post: all elements of this list equal to value have been
     // removed. The relative order of elements not equal to
     // value is unchanged. size has been updated correctly
     public void removeAll(Object obj){
          // Complete this method on the next page.
          //
          // Complete this method on the next page.
          //
          // Complete this method on the next page.
          //
          // Complete this method on the next page.
```

```
// pre: value != null
// post: all elements of this list equal to value have been
/ removed. The relative order of elements not equal to
// value is unchanged. size has been updated correctly
public void removeAll(Object obj){
```

4. (Using data structures, 15 points) Write a method that determines how many elements of a `SortedSet` are greater than a given value. The elements of the `SortedSet` are in ascending order. **This method is <u>not</u> in the `SortedSet` class so you do not have access to the `SortedSet`'s storage container or other instance variables.**

- Assume the `SortedSet` class in the question is generic based on Java's generic syntax.
- You may not use any other Java classes to solve this problem, except the `iterator` class, other classes' `equals` method and the `compareTo` method.
- The `SortedSet` is not altered as a result of this method.
- Recall, do not write code to check the precondition
- Your method must be O(1) <u>space</u>. In other words you cannot use an auxiliary array.

Examples:

```
().numGreater(0) -> returns 0
(1, 5, 10, 12, 50, 100).numGreater(0) -> returns 6
(1, 5, 10, 12, 50, 100).numGreater(1) -> returns 5
(1, 5, 10, 12, 50, 100).numGreater(12) -> returns 2
(1, 5, 10, 12, 50, 100).numGreater(200) -> returns 0
```

Here are the methods from the `SortedSet` class you may use:

`Iterator<C>` **`iterator`**`()`
Returns an iterator over the elements in this set in ascending order.

`boolean` **`add`**`(C obj)`
Adds the specified element to this set if it is not already present.

`boolean` **`contains`**`(Object o)`
Returns `true` if this set contains the specified element.

`int` **`size`**`()`
Returns the number of elements in this set (its cardinality).

`boolean` **`remove`**`(Object o)`
Removes the specified element from this set if it is present.

Recall the methods from the `Iterator` interface:

`boolean` **`hasNext`**`()`
Returns `true` if the iteration has more elements.

`C` **`next`**`()`
Returns the next element in the iteration.

`void` **`remove`**`()`
Removes from the underlying collection the last element returned by the iterator

Recall the `compareTo` method from the `Comparable` interface.

`int` **`compareTo`**`(Object o)`
Compares this object with the specified object for order. Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Complete the following method:

```
// pre: set != null, value != null, all elements of set are
// the same data type and that data type implements the Comparable
// interface, value is the same data type as the elements of set
public int numGreater(SortedSet set, Comparable value){
```

5. (Recursion, 15 points) The knapsack problem or "thief's dilemma" is this: Given a sack that has a weight limit and a set of items each of which has a value and a weight, determine which items should be put in the sack to maximize the value of the items in the sack.

For example, consider if we had the following items: (There is only one of each item.)

Weight in pounds	Value in dollars
12	4
2	2
2	1
1	1
4	10

and had a backpack that could hold 15 pounds maximum. (We are assuming the items are all very small and that we are not constrained by space, only weight. In other words if all the items together weighed less than the weight limit of the backpack they would all fit.)

The optimum solution for the above example is to take the all of the items expect the one that weighs 12 pounds and is worth 4 dollars.

Write a method that determines the maximum value that can be achieved given an `ArrayList` of `Items` and the maximum weight the backpack can hold. Note, your method does not need to determine which `Items` to include, only the maximum value that can be achieved. (This simplifies the problem somewhat.)

The `Item` class only has two methods:

`int getWeight()` Returns the weight of this item in pounds.
`int getValue()` Returns the value of this item in dollars.

Complete the following method: (There are no class, time, or space restrictions.)

```
// pre: things != null, capacity >= 0
// post: return the maximum possible value that can be obtained
// with the Items in the ArrayList named things and the given
// capacity. things is unaltered after this method completes.
public int getMaxValue(ArrayList<Item> things, int capacity){

          // Complete this method on the next page.
          //
          // Complete this method on the next page.
          //
          // Complete this method on the next page.
          //
          // Complete this method on the next page.
```

```
// pre: things != null, capacity >= 0
// post: return the maximum possible value that can be obtained
// with the Items in the ArrayList named things and the given
// capacity. things is unaltered after this method completes.
public int getMaxValue(ArrayList<Item> things, int capacity){
```

Scratch Paper

Name:_____

TAs name: Mikie Ron Sarah (Circle One)

Answer sheet for question 1, short answer questions

A. _____ I. _____

B. _____ J. _____

C. _____ K. _____

D. _____ L. _____

E. _____ M. _____

F. _____ N. _____

G. _____ O. _____

H. _____