Exam Number:

| Points off | 1 | 2 | 3 | 4 | | Total off | Net Score |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

# CS 307 – Midterm 2 – Fall 2010

Name__________________________________________

UTEID login name ________________________________

TA's Name: Harsh Yi-Chao (Circle One)

Instructions:

1. Please turn off your cell phones and other electronic devices.
2. There are 4 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(3)`?

```
public static int a(int x) {
      if(x <= 0)
            return 2;
      else
            return x * 2 + a(x - 1);
}
```

B. What will happen when the method call `b(5)` is made?

```
public static int b(int x) {
     if(x == 1)
          return 3;
     else
          return x + b(x - 3);
}
```

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

Recall the `substring` method:

```
public String substring(int beginIndex, int endIndex)

Returns a new string that is a substring of this string. The substring begins at
the specified beginIndex and extends to the character at index endIndex - 1.
Thus the length of the substring is endIndex-beginIndex.
```

```
public static String c2(String s) {
     String es = ""; // an empty String;
     String result = es;
     if(s.length() > 1) {
          if( s.charAt(0) == s.charAt(s.length() - 1) )
               result = es + s.charAt(0);
          result = c2(s.substring(1, s.length() - 1)) + result;
     }
     return result;
}

public static void c1(){
     String start = "AABBABBCA";
     System.out.print(c2(start));
}
```

D. What is returned by the method call `d(7)`?

```
public static int d(int x){
     if(x <= 2)
          return x * 2;
     else
          return 2 + d(x - 1) + d(x - 3);
}
```

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

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

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

```
public int f(int[] list, int tgt){
      int total = 0;
      for(int i = 0; i < list.length; i++)
            if(list[i] == tgt)
                  total++;
      for(int i = 1; i < list.length; i *= 2)
            total += list[i];
      return total;
}
```

G. What is the best case order (Big O) of method `g`? N = `list.size()`

```
public static int g(LinkedList<Double> list){
      int total = 0;
      for(int i = 0; i < list.size(); i++) {
            double hundreds = ((int) (list.get(i) * 100)) / 100;
            if(hundreds != 0 && hundreds != 5)
                  total += hundreds;
      }
      return total;
}
```

H. What is the worst case order (Big O) of method `h`? N = `words.size()`. Assume the `String` methods `length` and `charAt` are O(1).

```
public static void h(ArrayList<String> words, char ch) {
      int index = 0;
      while( index < words.size() ){
            String temp = words.get(index);
            if(temp != null && temp.length() > 0 && temp.charAt(0) == ch )
                  words.remove(index);
            else
                  index++;
      }
}
```

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

```
public static LinkedList<Double> i_ (double[] org){
    LinkedList<Double> result = new LinkedList<Double>();
    for(int i = 0; i < org.length; i++)
        result.add(0, org[i]); // 0 is the position to insert in the list
    return result;
}
```

J. What is the order (Big O) of method `j`? N = `mat.length`. `mat` is a square matrix.

```
// pre: mat is a square matrix
public static int j(int[][] mat) {
      int total = 0;
      for(int r = 0; r < mat.length; r++)
            for(int c = 0; c < mat.length; c++)
                  for(int i = 0; i < mat.length; i++)
                        total += i * mat[r][c];
      return total;
}
```

K. A sorting method uses the merge sort algorithm. It takes 19 seconds for the method to sort 500,000 distinct integers in random order. What is the expected time for the method to sort 1,000,000 distinct integers in random order?

L. A method is O($N^3$). It takes the method 2 seconds to run when N = 10,000. What is the expected time for the method to run when N = 30,000

M. A method is O(N). When N = 100,000 the method takes 10 seconds to run. Given 0.2 seconds, how many pieces of data can the method process?

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

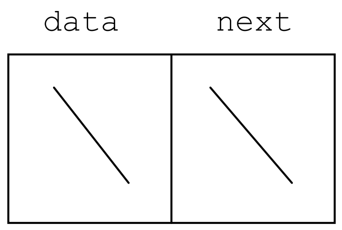
```
public class Node {
     public Object data;
     public Node next;

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

N. What is output by the following code?

```
Node n1 = new Node(2, null);
n1.next = new Node(7, new Node(5, n1));
Node n2 = n1.next;
n2 = n2.next.next;
System.out.print(n2.data);
```

O. Draw the resulting variables and objects after the following code executes. Draw node objects as shown below. (The example has all 2 instance variables set to `null`. Any variables that refer to the example below are not shown. You must show them in your drawing.) Use arrows to show the references that exist and a forward slash to indicate variables that store `null`.



```
Node n3 = new Node("A", null);
Node n4 = new Node(n3, n3);
n3.next = new Node(null, n4);
```

2. (Data Structures 25 points) Write a method for the `AbstractSet` class from assignment 8 that removes all of the elements present in an `ISet` passed as a parameter from the calling object.

- 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.
- 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.
- The calling object may be altered as a result of this method. The parameter `other` is not altered as a result of this method call.

Examples:

```
(2, 1, 6, 4).removeAll( (4, 3, 2, 6) ) -> calling object becomes (1)
(1, 2).removeAll( (2, 1) ) -> calling object becomes ()
(1, 5, 2, 3).removeAll( () ) -> calling object remains (1, 5, 2, 3)
().removeAll( () ) -> calling object remains ()
(4, 1, 2, 15).removeAll( (12, 5, -1, 7) ) -> calling object remains
     (4, 1, 2, 15)
```

`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 this method is part of the `AbstractSet` class.

```
// pre: other != null
// post: Remove all elements that are in other from this set.
//   other is not altered as a result of this method call.
public void removeAll(ISet<E> other) {
```

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

// pre: other != null
// post: Remove all elements that are in other from this set.
//   other is not altered as a result of this method call.
public void removeAll(ISet<E> other) {
```

3. (Linked Lists 25 points). Write a method for a `LinkedList` class that determines if the elements stored in the list are *strictly decreasing* according to the natural ordering of the elements. A list is strictly decreasing if each element is less that the element before it.

- The list uses singly linked nodes that store one piece of data and a reference to the next node in the list.
- The list only maintains a reference to the first node in the linked structure.
- If the list is empty the reference to the first node is set to `null`.
- The list does not maintain a size instance variable.
- The last node's next reference is set to `null`.

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.

Here are some examples and the expected results. For these examples assume the values shown are `Integer` objects.

```
[].strictlyDecreasing() -> returns true
[5].strictlyDecreasing() -> returns true
[5, 3].strictlyDecreasing() -> returns true
[5, 3, 1, -1].strictlyDecreasing() -> returns true
[5, 5, 1, -1].strictlyDecreasing() -> returns false
[5, 1, 1, -1].strictlyDecreasing() -> returns false
[6, 4, 12, 15, 3].strictlyDecreasing() -> returns false
```

Here is the Node class:

```
public class Node<E extends Comparable<? super E>> {
     public Node(E value, Node<E> next)

     public E getData()
     public Node<E> getNext()

     public void setValue(E value)
     public void setNext(Node<E> next)
}
```

Note the generic syntax declaration for the class means the data in `Nodes` are guaranteed to implement the `Comparable` interface.

Recall the `compareTo` method that may be called on objects of type `Comparable`.

`int` **`compareTo`**`(E obj )` Compares this object with `obj` for order.

**Returns:** a negative integer if this object is less than `obj`, 0 if this objects equals `obj`, or a positive integer if this object is greater than `obj`.

```
public class LinkedList<E extends Comparable<? super E>> {

     private Node<E> first;

// Complete the following method:

// pre: none
// post: return true if the elements in this LinkedList are
//   strictly decreasing, false otherwise
public boolean isStrictlyDecreasing() {
```

4. (Recursion, 20 points) A very precise golfer is trying to determine the minimum number of times he can a hit a ball and reach a target. The distance to the hole is given as an int. The golfer has a choice of clubs. The golfer is *very precise* and when using a particular club always hits the ball the exact same distance. The golfer doesn't use traditional names for his clubs. (e.g. driver, 3 wood, 5 iron) Instead he names each club based on the exact distance he can hit the ball with that club. (e.g. 10, 40, 60).

The golfer may use a particular club as many times as he wants. For this question assume the golfer always hits the ball straight. Further assume the golfer DOES NOT want to consider scenarios where he hits the ball past the target (overshoots) and then turns around to hit the ball back to the target.

You will complete the following method:

```
// pre: clubs != null, all elements of clubs > 0
// post: Return the minimum number of times the golfer must hit a
// ball to reach the given distance or -1 if it is not possible to
// reach the given distance with the given clubs.
public int minHits(int distance, int[] clubs) {
```

Here are some examples. The explanations in parenthesis are for illustrative purposes. They are not returned by the method.

```
minHits(-30, {30, 10, 50}) -> -1   (not possible)
minHits(0, {30, 10, 50}) -> 0   (already there)
minHits(15, {30, 10, 50}) -> -1   (not possible)
minHits(50, {30, 10, 50}) -> 1   (use the 50 once)
minHits(70, {30, 10, 50}) -> 3   (use the 30, use the 30, use the 10)
minHits(105, {40, 10, 60, 200}) -> -1   (not possible)
minHits(190, {40, 10, 60, 200}) -> 4   (60, 60, 60, 10)
minHits(90, {40, 10, 60, 200}) -> 3   (40, 40, 10)
minHits(90, {15}) -> 6   (15, 15, 15, 15, 15, 15)
```

Complete the method on the next page:

Complete the method on the next page:

Complete the method on the next page:

```
// pre: clubs != null, all elements of clubs > 0
// post: Return the minimum number of times the golfer must hit a
// ball to reach the given distance or -1 if it is not possible to
// reach the given distance with the given clubs.
public int minHits(int distance, int[] clubs) {
     assert clubs != null
```

Name:________________________________

TA's Name: Harsh Yi-Chao (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: