Points off	1	2	3	4	5	6	Total off	Net Score

CS 307 – Final – Fall 2009

Name_____

UTEID login name _____

Instructions:

1. Please turn off your cell phones.
2. There are 6 questions on this test.
3. You have 3 hours to complete the test.
4. You may not use a calculator on the test.
5. You may add helper methods if you wish when answering coding questions.
6. When answering coding questions assume the preconditions of the methods are met.

1. (1.5 point each, 30 points total) Short answer. Place you answers on the attached answer sheet.

For questions that ask what is the output:

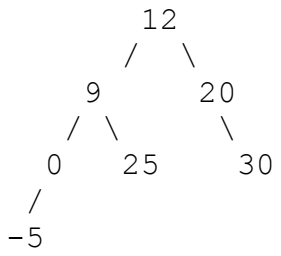
- If the code contains a syntax error or other compile error, answer "Compiler error".
- If the code would result in a runtime error or exception answer "Runtime error".
- If the code results in an infinite loop answer "Infinite loop".

On questions that ask for the Big O of a method or algorithm, recall that when asked for Big O your answer should be the most restrictive, correct Big O function. For example Selection Sort has an expected 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 $O(N^3)$. Give the most restrictive, correct function. (Closest without going under.)

A. The following values are inserted one at a time in the order shown into an initially empty binary search tree using the traditional, naïve insertion algorithm. Draw the resulting tree.

```
72  125  218  6  51  41
```

Consider the following binary tree. `12` is the root of the tree.



B. What is the result of a pre-order traversal of the binary tree shown above?

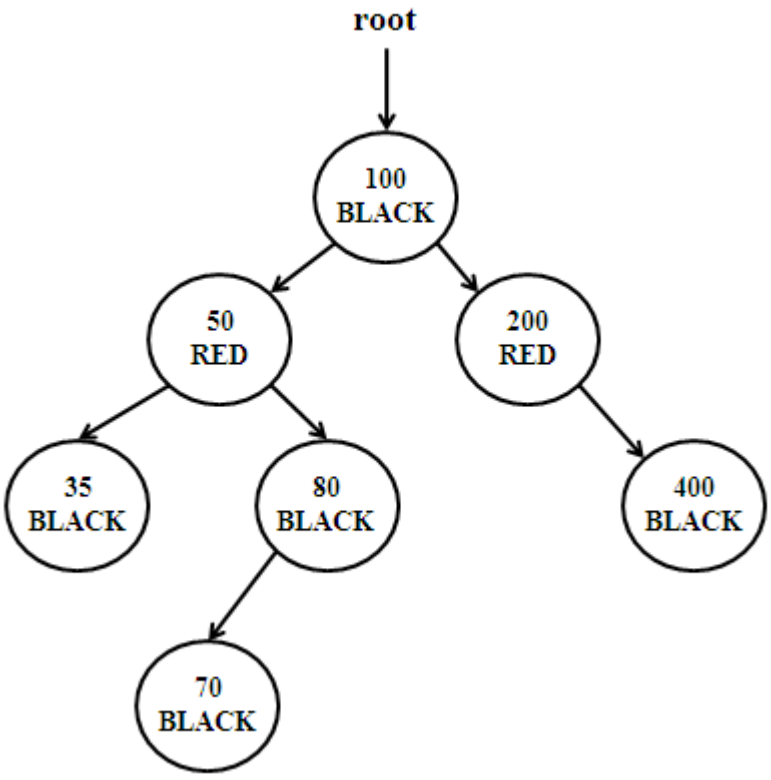
C. What is the result of a in-order traversal of the binary tree shown above?

D. What is the result of a post-order traversal of the binary tree above?

E. Is the tree shown above a binary search tree?

F. On the answer sheet, fill in the nodes of the given binary tree with integer values between 1 and 10 to make the tree shown a binary search tree.

G. Explain why the following tree is not a red black tree. The values in each node are integers. The color of the node is below the value.



H. Without moving or rotating any nodes is it possible to make the tree part G a red black tree? In other words you can only change the color of nodes.

Consider the following method:

```
public int sumList(List<Integer> list) {
    int total = 0;
    for(int i = 0; i < list.size(); i++)
        total += list.get(i);
    return total;
}
```

I. What is the Big O of method `sumList` if `list` is a Java `ArrayList`?

J. What is the Big O of method `sumList` if `list` is a Java `LinkedList`?

K. Is method `myst`, shown below, functionally equivalent to method `sumList`? In other words, given the same list does it always return the same answer as method `sumList`? Briefly explain why or why not.

```
public int myst(List<Integer> list) {
    int val = 0;
    for(int x : list)
        val += x;
    return val;
}
```

L. What is the Big O of the following method? The `BST` class uses a Red-Black tree as its internal storage container.

```
public BST<Integer> buildTree(int N){
    BST<Integer> result = new BST<Integer>();
    for(int i = 0; i < N; i++)
        result.add(i);
    return result;
}
```

M. Write an infix expression that is equivalent to the following postfix expression. (Don't evaluate the expression. Write an infix expression that contains all the same operands and operators.)

```
3 2 + 5 2 + *
```

N. What is output by the following code?

```
int[] data = { 13, 15, 1, 15, 14, 4, 8, 4};
Stack<Integer> st = new Stack<Integer>();
for(int x : data)
    st.push(x);
while(!st.isEmpty()){
    int val = st.pop();
    if( val % 4 != 0 )
        System.out.print(val + " ");
}
```

O. What is output by the following code? (The Queue class is not the Java Queue interface, rather it is a class that implements a traditional Queue.) The method `isVowel` returns `true` if the parameter is equal to any of the following characters: a, A, e, E, i, I, o, O, u, U

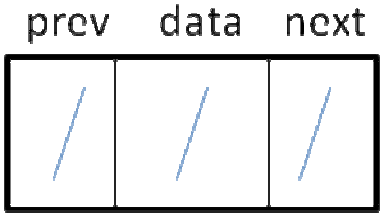
```
Queue<Character> hold = new Queue<Character>();
String word = "zoosporous";
for(int i = 0; i < word.length(); i++)
    if( !isVowel(word.charAt(i)) )
        hold.enqueue(word.charAt(i));

while(!hold.isEmpty())
    System.out.print(hold.dequeue() + " ");
```

P. Consider the following Node class:

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

    public Node(Object o, Node p, Node n) {
        data = o; prev = p; next = n;
    }

    public Node(){
        data = null; prev = null; next = null;
    }
}
```

Draw the resulting variables and objects after the following code executes. Draw node objects as shown below. (The example has all three instance variables set to `null`.) Use arrows to show the references that exist and a forward slash to indicate variables that store `null`.

```
Node n1 = new Node();
n1.prev = n1;
Node n2 = new Node(null, n1, n1);
Node n3 = new Node(null, n2, n1);
n1 = n2;
```



Q. Suppose you want to encode 120 different colleges using a fixed length encoding scheme. (Each college is represented with a unique combination of bits that is the same length as all of the other college codes.) You do not need to store any other information about the college. You simply need a unique identifier for each college. What is the minimum number of bits per college required to have a unique code for each college?

R. 1000 distinct (no repeats) integers that are in random order are inserted one at a time into a binary search tree using the traditional algorithm presented in class. What is the expected height of the resulting tree? Give the actual number as you did on the experiments in assignment 11, NOT the Big O of the height.

S. Give an example of one operation that is more efficient when using a circular, doubly linked list as opposed to an array based list.

T. Briefly explain the purpose of the identifier `E` in the following class header:

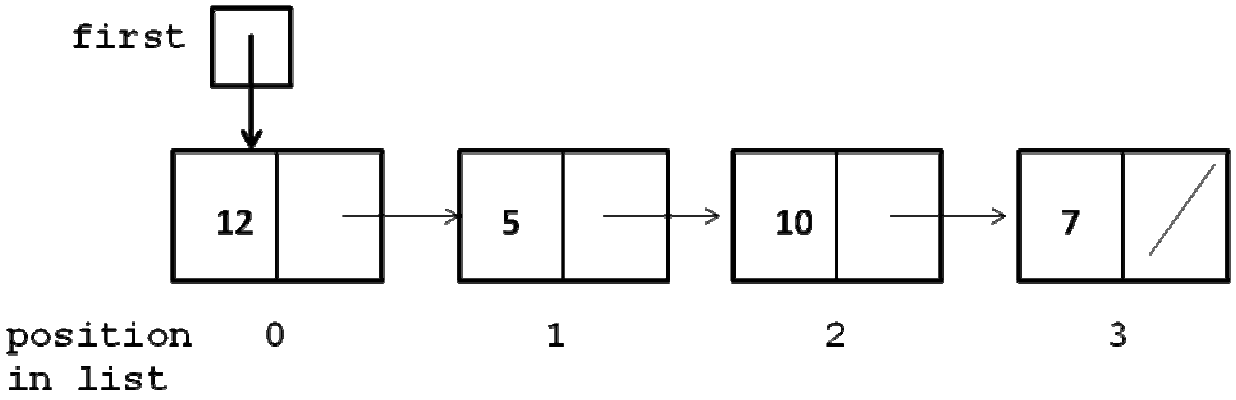
```
public class LinkedList<E> {
```

2. (Linked Lists, 15 points) Implement an instance method for a linked list class that splits the list as described below.

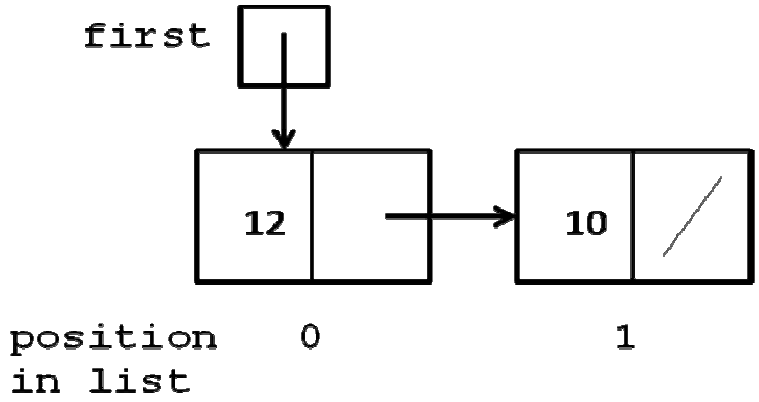
The LinkedList class has the following properties:

- The nodes are singly linked.
- The LinkedList class keeps a reference to the first node in the linked structure, named `first`.
- The last node in the linked structure has its next reference set to `null`.
- If the linked list is empty the reference to the first node is set to `null`.
- The linked list does not have an instance variable for its size.
- The linked list class achieves genericty by using Java's generics.

Consider the following linked list and its internal data structure. The linked list in this example contains Integers.



Complete a method named `split` that removes all elements from the list that were initially at odd positions. (0 is considered an even number.) For example, if the list had the initial configuration shown above the resulting list would be:



Imporant: Your method must be O(1) space meaning you can not use an auxillaury native array, ArrayList, or other data structure. Your method should be O(N) time where N is the number of elements initially in the LinkedList. You may not use any other methods from the LinkedList class unless you write them yourself on this question.

Here are some more examples of calls to `split`.

```
[].split() results in [] (split has no effect on an empty list)
[12].split() results in [12] (no effect on a list of size 1)
[12, 5].split() result in [12]
[12, 5, 6].split() results in [12, 6]
```

Here is the `Node` class the linked list class uses:

```
public class Node<E>{
    public Node(E value, int priority, Node<E> next);

    public Node<E> getNext();
    public int getPriority();
    public E getValue();

    public void setNext(Node<E> newNext);
}
```

Here is the `LinkedList` class.

```
public class LinkedList<E>{

    // Reference to the front node of the linked list.
    // If this LinkedList is empty first == null
    private Node<E> first;

    // complete the following method:
    // pre: none
    public void split(){
```

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

COMPLETE THIS METHOD ON THE NEXT PAGE

COMPLETE THIS METHOD ON THE NEXT PAGE

COMPLETE THIS METHOD ON THE NEXT PAGE

```
// instance method in the LinkedList class
// pre: none
public void split(){
```

3. (Using Data Structure, 20 points) Write a method that determines if an input String that contains grouping symbols and other characters is balanced. For this question grouping symbols are characters that come in pairs. Each pair has an opening symbol and a closing symbol. There are three pairs of grouping symbols in this questions:

open	close
(	)
{	}
<	>

Grouping symbols are balanced if every opening symbol has a corresponding closing symbol and all pairs are correctly nested. In other words if the last opening symbol was '{' then the next grouping symbol must either be an opening symbol (of any type) or the appropriate closing symbol. In this case '}'.

In this question a character that is not a grouping symbol has no effect on whether the String is balanced or not. Here some examples of Strings that are balanced:

```
""
"x"
"()"
"(x)"
"(<x>y)"
"<x<y{(z)x(y)}>z>"
```

Here are some examples of String that are not balanced:

String	Reason no balanced
"x{"	The { does not have a matching }.
"<<>}"	The leftmost < does not match the }.
"<{}>)"	The) does not have an opening (.

Write a method that returns `true` if a given String is balanced as described above, `false` otherwise. The only data structure you can use is a Stack and the String methods mentioned below. Do not use recursion on this question.

Assume you have a Stack class available with the following methods:

```
public class Stack<E> {
      public Stack() // creates a new, empty Stack
      public void push(E data)
      public E top()
      public E pop()
      public boolean isEmpty()
}
```

The method you are writing is in a class with two String instance variables.

```
private static String OPEN =  "({<";
private static String CLOSE = ")}>";
```

Note, the Strings are parallel. The position of the opening symbol in `OPEN` matches the position of the closing symbol in `CLOSE`.

The String methods you may use are:

`char` **`charAt`**`(int index)` Returns the `char` value at the specified index.

`int` **`indexOf`**`(int ch)` Returns the index within this string of the first occurrence of the specified character or -1 if ch is not present in this String.

`int` **`length`**`()` Returns the length of this string.

Complete the following method:

```
private static String OPEN =  "({<";
private static String CLOSE = ")}>";

// pre: expr != null
// post: return true if the grouping symbols in expr are balanced
//       false otherwise
public boolean balanced(String expr){
    assert expr != null;
    Stack<Character> st = new Stack<Character>();
    // recall chars can be pushed onto st directly via autoboxing
```

```
// more room on next page in needed
```

4. (Binary Search Trees, 10 points) Complete a method that determines the range of values stored in a binary search tree of integers. The range of values equals the maximum value in the binary search tree minus the minimum value. If there is one value in the tree the range is 0. If the tree is empty the range is undefined. You may not use any other data structures on this question. You may not use any other methods from the `BinarySearchTree` class unless you write those methods yourself on this question.

```
public class BinarySearchTree {

     // when size == 0, root == null
     private BSTNode<Integer> root;
     private int size;
```

Here is the public interface of the `BSTNode` class.

```
public class BSTNode<E extends Comparable<E>> {
    public BSTNode()
    public BSTNode(E initValue)
    public BSTNode(E initValue, BSTNode<E> left, BSTNode<E> right)

    public E getValue()
    public BSTNode<E> getLeft()
    public BSTNode<E> getRight()

    public void setValue(E theNewValue)
    public void setLeft(BSTNode<E> theNewLeft)
    public void setRight(BSTNode<E> theNewRight)
}
```

Complete the following method in the BinarySearchTree class:

```
// pre: size() > 0
// post: returns the range of the data in this BinarySearchTree
//       This BinarySearchTree is not altered.
public int range() {
    assert size() > 0;
```

```
// more room on next page in needed
```

5: (Binary Trees, 10 points) Write a method to determine how many times a given value appears in a binary tree. (Not a binary search tree, a plain binary tree. Duplicates are possible.) You may not use any other data structures on this question. You may not use any other methods from the `BinaryTree` class unless you write those methods yourself on this question.

```
public class BinaryTree<E> {

     // when size == 0, root == null
     private BNode<E> root;
     private int size;
```

Here is the public interface of the `BNode` class.

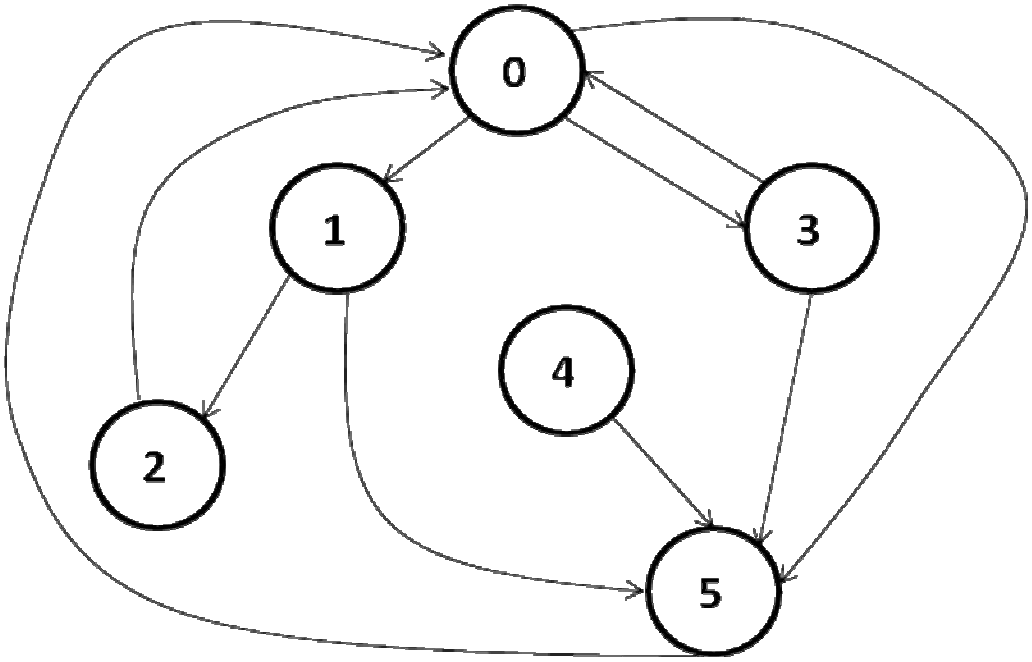
```
public class BNode<E> {
    public BNode()
    public BNode(E initValue)
    public BNode(E initValue, BNode<E> left, BNode<E> right)

    public E getValue()
    public BNode<E> getLeft()
    public BNode<E> getRight()

    public void setValue(E theNewValue)
    public void setLeft(BNode<E> theNewLeft)
    public void setRight(BNode<E> theNewRight)
}
```

Complete the following method in the BinaryTree class:

```
// pre: obj != null
// post: returns the number of times obj is present in this
//       BinaryTree. This BinaryTree is not altered as a result of
//       this method call.
public int numPresent(E obj) {
    assert obj != null;
```

```
// more room on next page in needed
```

6. (New data structures, 15 points) A *graph* is a data structure that consisting of vertices (which are like nodes) and edges. (which are like links) In a directed graph links contain an arrow indicating if movement is allowed from one node to another. Binary trees are an example of a directed acyclical graph, meaning no cycles are present. In this question the graph may contains cycles. (A cycle is like a loop.) Here is an example of a graph. The nodes are number 0 to N - 1 where N is the number of nodes in the graph. The data stored in the nodes are not shown in this example. (It assumed that movement is always allowed from a node to itself.)



One way of representing a graph is via an adjacency matrix. The adjacency matrix is a square 2d array of booleans with N rows and N columns. If there is a direct link from the node with index *row* to the node with index *column* then the adjacency matrix stores true, otherwise it stores false. The graph above would be represented with the following adjacency matrix:

	0	1	2	3	4	5
0	true	true	false	true	false	true
1	false	true	true	false	false	true
2	true	false	true	false	false	false
3	true	false	false	true	false	true
4	false	false	false	false	true	true
5	true	false	false	false	false	true

Write a method `pathExists` that returns true if a path exists from a start node to an end node.

```
public boolean pathExists(int start, int end)
```

If g is the graph shown above here are some examples of calls it `pathExists`:

```
g.pathExists(0, 1) -> true
g.pathExists(0, 2) -> true
g.pathExists(0, 3) -> true
g.pathExists(0, 4) -> false
g.pathExists(0, 5) -> true
g.pathExists(1, 4) -> false
g.pathExists(1, 5) -> true
```

Here is the `Graph` class:

```
// represents a directed graph with N nodes
// The graph will always contain at least 2 nodes
public class Graph{

    // adjMatrix is a square matrix. adjMatrix.length equals the
    // number of nodes in this graph
    private boolean[][] adjMatrix;

     public int numNodes() {
          return adjMatrix.length;
     }


    // pre: 0 <= start < numNodes, 0 <= end < numNodes(),
    //      start != end
    // post: return true if a path exists from start to end
    //       This Graph is not altered as a result of this method.
    public boolean pathExists(int start, int end) {
        // Complete the method on the next page.
```

Hints:

- Your method does not need to find the shortest path, just whether a path exists or not.
- There are two common approaches to this problem:
    - Recursive backtracking. This is a very typical recursive backtracking problem.You may assume the graph is small enough that your code will not suffer a stack overflow as long as you do not have a logic error.
    - The same approach you used on the Word Ladder assignment, which will find the shortest path. (The major difference being this question deals with a directed graph and the words on the word ladder assignment were an undirected graph.)
- You may use other data structures if you wish. Lists, Stacks, Queues. An, `ArrayList` of `Integer`s than contains the nodes you have already visited would be very helpful.
- If you take the recursive backtracking approach it would be extremely helpful to have a helper method.

COMPLETE THE METHOD ON THE NEXT PAGE.

COMPLETE THE METHOD ON THE NEXT PAGE.

COMPLETE THE METHOD ON THE NEXT PAGE.

```
// pre: 0 <= start < numNodes, 0 <= end < numNodes(),
//      start != end
// post: return true if a path exists from start to end
//       This Graph is not altered as a result of this method.
public boolean pathExists(int start, int end) {
    assert 0 <= start && start < numNodes() && 0 <= end
        && end < numNodes && start != end;
```

Name:_____

Answer sheet for question 1, short answer questions. **Put answers next to the letter.**

_____

A:

_____

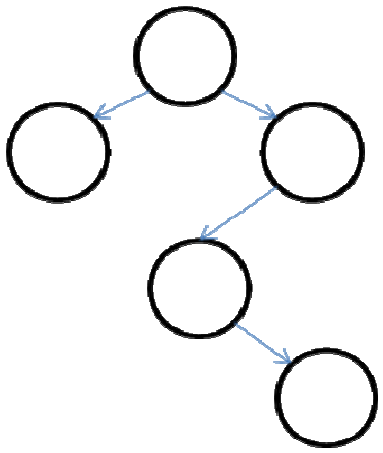
B. _____

C._____

D._____

E._____

F.



_____

G._____

H._____

I._____

J._____

K._____

L._____

M._____

N._____

O._____

P._____

Q._____

R._____

S._____

T._____