CS 307 – Final Exam – Spring 2002

Name____________________________________

Last 4 digits of SSN / Student ID ______________

Class Unique ID ___________________________

Instructions:
1. There are 4 questions on this test. Question 1 is worth 60 points, all others are worth 20 points each.
2. You will have 3 hours to complete the test.
3. You may not use a calculator.
4. When code is required, write Java code.
5. You may not use any classes or methods from the Java Standard Library other than the ones specifically mentioned on each question.
6. The style guide is not in effect.
7. Please make your answers legible.

1. Java mechanics and short answer questions. (2 points each) Write the answer to each question in the space provided. If code results in an error indicate if it is a compile error or runtime error.

A. The following numbers are inserted in the order shown into a binary search tree with no checks to ensure or maintain balance. The tree is initially empty. Draw the resulting tree.

307 310 108 408 313 336 105
For parts B, C, and D consider the following binary tree. For each question assume when a node is processed the value in the node is printed out by the statement:

```java
System.out.print( currentNode.data + " ");
```

![Binary Tree Diagram]

B. What is the output of a preorder traversal of the tree?

____________________________________________________

C. What is the output of an inorder traversal of the tree?

____________________________________________________

D. What is the output of a postorder traversal of the tree?

____________________________________________________
E. Is the tree on page 2 a binary search tree?

______________________

F. Consider inserting 1 item into a binary search tree that contains N nodes. There are no special algorithms to ensure or maintain balance in the tree as items are inserted or deleted. What is the average case Big O for inserting 1 item into the tree?

______________________

G. Consider inserting 1 item into a binary search tree that contains N nodes. There are no special algorithms to ensure or maintain balance in the tree as items are inserted or deleted. What is the worst case Big O for inserting 1 item into the tree?

______________________

H. Consider inserting N items into a binary search tree that is initially empty. There are no special algorithms to ensure or maintain balance in the tree as items are inserted or deleted. What is the average case Big O for inserting N items into the (initially empty) tree?

______________________

I. Consider inserting N items into a binary search tree that is initially empty. There are no special algorithms to ensure or maintain balance in the tree as items are inserted or deleted. What is the worst case Big O for inserting N items into the (initially empty) tree?

______________________

J. What is the Big O of this code? method1 always has a Big O of O(1).

```c
for (int i = 0; i < N; i++)
    for (int j = 0; j < i; j++)
        for (int k = 1; k < j; k++)
            method1();
```

______________________
K. What is the Big O of this code? `method` always has a Big O of O(N) where N is the magnitude of the parameter sent to `method2`.

```java
for (int i = 0; i < N; i++)
    for (int j = N; j > 0; j /= 2)
        method2(j)
```

L. What is the Big O of this code? `method3` always has a Big O of O(N) where N is the magnitude of the argument sent to `method3`. `method4` has a Big O of O(log N) where N is the magnitude of the parameter sent to `method4`.

```java
for (int i = 0; i < N; i++)
    method3(i);
for (int j = 0; j < 3 * N / 4; j++)
    method4(j);
for (int k = 0; k < N; k++)
    method3(k);
```

M. What is the Big O of the following method?

```java
public static int method5(int N)
{
    if (N < 0)
        return 5;
    else
        return 2 + method5(N - 1);
}
```

N. What is the average case Big O of a preorder traversal on a binary search tree?
O. A **full** binary tree is one where every level has the maximum number of nodes. How many nodes are in a full binary tree with 6 levels?

_______________________________________

P. Consider the following method:

```java
public int OS( int n )
{
    if ( n <= 1 )
        return n + 4;
    else
        return n + OS( n / 3 );
}
```

What is the output of the statement `System.out.println( OS(41) );`?

_______________________________________

Q. Consider the following ListNode class:

```java
public class ListNode
{
    public Object data;
    public ListNode next;
    public ListNode prev;
}
```

Draw the object variables and objects that exists after the following code is executed. Use a "/" to indicate any object references that are null.

```java
ListNode N1 = new ListNode();
ListNode N2 = new ListNode();
N1.next = N2;
N2.prev = N1;
ListNode N3 = new ListNode();
```

_______________________________________

R. A program with a Big O of O(N) takes 10 seconds to process a data set of size 200,000. What is the expected time for the same program to process a data set of 500,000 items?

_______________________________________
S. A program with a Big O of \(O(N^2)\) takes 10 seconds to process a data set of size 50,000. What is the expected time for the same program to process a data set of 200,000 items?

T. Consider the following code:

```java
int[] list; // line 1

// code to create and fill list not shown
// you may assume list is not null

list[12] = 0; // line 2
```

Can line 2 cause an exception? If so what must be done so that an exception cannot occur?
2. (Trees) This question asks you to write a method that examines a binary tree. Recall that each node in a binary tree has a height. A node's height is equal to the path length from the node to its deepest leaf. The path length is equal to the number of links that must be navigated to get from one node to another.

You are to write a method that determines how many nodes would have to be added to each level to make it full. A full level is one that has the maximum number of nodes possible. Consider the binary tree on page 2. Level 0 is short 0 nodes (this will always be the case for a non-empty tree), level 1 is also 0 nodes short, level 2 is 1 node short, and level 3 is 5 nodes short. You may create helper methods if you wish.

You may use the following method:

```java
/* pre: none
post: return height of tree for which n is the root
returns -1 if n = null
Big O is O(N) where there are N nodes in the tree */

public static int getHeight(BTNode n)
```

The method you write and `getHeight` use the following `BTNode` class:

```java
public class BTNode {
    public Object data;
    public BTNode left, right;
}
```

```java
/* pre: none
post: return an array of ints with the number of nodes missing from each level in the tree for which n is the root. The number of missing nodes is how many nodes the level would need to be full. The length of the returned array shall equal the depth of the tree + 1. Each element in the array shall correspond to the level of the tree. If n = null, null is returned
The Big O of this method is not to exceed O(N) where there are N nodes in the tree */

public static int[] nodesMissing(BTNode n) {
    // complete this method below. You may create helper methods
```
/* pre: none
    post: return an array of ints with the number of nodes missing from each level in the tree for which n is the root. The number of missing nodes is how many nodes the level would need to be full. The length of the returned array shall equal the depth of the tree + 1. Each element in the array shall correspond to the level of the tree. If n = null, null is returned
    The Big O of this method is not to exceed O(N) where there are N nodes in the tree
*/

public static int[] nodesMissing(BTNode n)
{
    // complete this method below. You may create helper methods
3. (Recursion) Write a method to determine how much space is taken up by the files in a directory.

This question uses 2 classes called Directory, and File. The public interface for the classes are shown below. Directories themselves do not take up any space.

```
public class Directory
{
    /* pre: none
    post: returns an array of Directories in this Directory. The returned array has been trimmed so that all elements are non null and contain Directory objects or if there are no Directories in this Directory, null is returned.
    */
    public Directory[] getDirectories()

    /* pre: none
    post: returns an array of Files in this Directory. The returned array has been trimmed so that all elements are non null and contain File objects, or if there are no Files in this Directory, null is returned.
    */
    public File[] getFiles()
}

public class File
{
    /* pre: none
    post: returns the size of this file in kilobytes
    */
    public int getSize()
}
```

Complete the following method. You may create a helper method if you wish.

```
/* pre: dir != null
post: returns the total space used by all files that are part of this directory (including all files in all subdirectories)
*/
public int getSize(Directory dir)
{
    // complete this method on the next page.
}
/*    pre: dir != null
    post: returns the total space used by all files that are part
          of this directory (including all files in all subdirectories)
*/

public int getSize(Directory dir)
{ // complete this method below
4. (Using data structures) Implement a method to convert an infix expression to a postfix expression. 

The algorithm for the conversion is as follows:

Examine the stream of tokens that make up the infix expression. Perform the following action for each token type: Everything popped from the stack is immediately added to the postfix expression unless it is an open parenthesis.

Operands (numbers or variables): Immediately add to the postfix expression

Close parenthesis: Pop all stack symbols until an open parenthesis occurs. Neither the open or close parenthesis are added to the postfix expression.

Operators (including open parenthesis): Pop the stack until an operator of lower precedence OR a right associative operator of equal precedence appears at the top of the stack OR the stack is empty. Then push the operator.

End of Input: (No more tokens.) Pop all remaining stack symbols.

You will use the following classes for this question.

```java
public class Stack {
    public Stack() //create an empty stack
    public void push(Object x)
    public void pop() //remove top element
    public Object top() //access top element without removing it
    public void makeEmpty()
    public boolean isEmpty()
    public int size()
}
```

```java
public class ExpressionToken {
    public static int OPERAND = 0;
    public static int CLOSE_PAREN = 1;
    public static int OPEN_PAREN = 2;
    public static int OPERATOR = 3;

    /* returns an int corresponding to the type of Token this is, either OPERAND, CLOSE_PAREN, OPEN_PAREN or OPERATOR */
    public int getType()

    /* pre: getType = OPERATOR
        post: returns true if this token is right associative */
    public boolean isRightAssociative()
```
// ExpressionToken class continued

/* pre: getType = OPERATOR
post: returns the precedence of this operator, value will
be > 0. The higher the precedence of the operator the
larger the int returned.
*/
public int getPrecedence()
} // end of ExpressionToken class

public class Expression
{
    // default constructor. Creates an expression with no terms
    public Expression()

    /* pre: none
    post: returns an ExpressionIterator that can be used to
    examine all the terms in this Expression. The
    ExpressionIterator is positioned on the first term in the
    Expression.
    */
    public ExpressionIterator getIterator()

    /* pre: et != null
    post: et added to this expression as the last term or
    operator
    */
    public void addExpressionToken(ExpressionToken et)
} end of Expression class

public class ExpressionIterator extends Iterator
{
    /* pre: hasNext = true
    post: returns the current ExpressionToken and advances to
    the next ExpressionToken if it exists
    */
    public ExpressionToken next()

    /* pre: none
    post: returns true if there is at least one more
    ExpressionToken in the Expression this Iterator is
    examining.
    */
    public boolean hasNext()
} end of ExpressionIterator class

Complete the method createPostfixExpression on the next page. This method is not a member method of any of the classes listed above.
/** pre:  ex != null, ex is a valid infix expression 
   post: return an Expression object that is a prefix expression 
       equivalent to ex. 
*/
public Expression createPostfixExpression(Expression ex) 
{   // complete this method below. You may create helper methods