Points off 1 2 3 4 5 Total off Net Score

CS 307 – Midterm 2 – Summer 2002

Name_______________________________

Last 4 digits of SSN / Student ID ____________

Class Unique ID _________________________

Instructions:
1. There are 4 questions on this test.
2. You will have 3 hours to complete the test.
3. You may not use a calculator on the test.
4. When code is required, write Java code.
5. The style guide is not in effect.
6. Ensure your answers are legible.
7. When writing code you may not use any methods or classes from the Java Standard Library except as noted and native arrays including the length constant, the System.out.print, System.out.println, and the equals method.

1. (2 points each, 20 points total) Java Mechanics and theory. If an error would occur answer "syntax error" or "runtime error" depending on what type of error it would be.

A. What is the average case Big O for finding the maximum value in a native array of unsorted integers?

B. Assume you are implementing a quicksort algorithm. You choose to use the median of three method to find the pivot. To do this your code will look at the first, last, and middle element of the unsorted list and pick the median of those three as the pivot. The indices for the beginning and end of the list are known. What is the Big O of the finding the median of three?
C. Consider the following code from class M2:

```java
public static int stone(int n) {
    if(n <= 0)
        return 3;
    else
        return n + stone(n - 2);
}
```

What is printed out by the following code?

```java
System.out.println( M2.stone(8) );
```

D. Consider the following code from class M2:

```java
public static int mav(int m) {
    if(m <= 0)
        return 2;
    else
        return 2 + mav(m - 1) + mav(m - 2);
}
```

What is printed out by the following code?

```java
System.out.println( M2.mav(5) );
```

E. Given a positive integer N, what is the Big O of M2.mav( N ) from question 1.D?
F. Consider a stack of integers. A stack has one access point, the top. The push operation adds the parameter listed to the top of the stack. The pop operation removes the top element. The top operation returns a copy of the top element without removing it. Draw a picture of the stack after the following code is executed. Clearly label the top of the stack.

```java
Stack s = new Stack();
s.push(12);
s.push(9);
s.pop();
s.push(73);
s.push(81);
s.pop();
s.push(55);
s.push(s.top());
s.push(10);
```

G. Consider the following code:

```java
for (int i = 0; i < N; i++)
    for (int j = 1; j < N; j *= 2)
        west(i);
```

Method `west` has a Big O of O(N), where N is the magnitude of the parameter sent to the method. What is the Big O of the above code?
H. Consider the following code from class M2:

```java
public static int warrior(int n)
{
    if(n == 0)
        return 3;
    else
        return n + warrior(n - 2);
}
```

What is printed out by the following code?

```java
System.out.println( M2.warrior(7) );
```

---

I. Consider a stack that stores integers. The stack operations are the same as described in question 1.F. What is output by the following code?

```java
Stack s = new Stack();
s.push( 12 );
s.push( 2 );
s.push( 8 );
s.push( 31 );
s.pop();
s.push( s.top() );
s.push( s.top() );
for(int i = 0; i < 3; i++)
    s.pop();
System.out.println( s.top() );
```

---

J. Rank the following 3 sorts from slowest to fastest in terms of average case Big O.

shell sort    selection sort    quick sort
2. Recursion (25 points) Flood fill. Write a method to simulate the flood fill option that many graphics programs provide. A window or drawing is represented as a 2D array of integers where the integers represent different colors. The floodFill method will be passed a 2D array of integers as well as the row and column to start filling from and the color to change to. The method shall change the color of the specified element to the new color, as well as any elements connected to the original element via a move of one up, down, left, right, or diagonally, that are the same color as the original element, and then apply these rules to those adjacent elements as well.

/* pre: canvas != null, canvas is a rectangular matrix. (all rows have the same number of columns)
   0 <= row < canvas.length, 0 <= col < canvas[0].length
   post: flood fill performed as explained above. */

public void floodfill(int[][] canvas, int row, int col, int newColor)

If canvas is equal to the matrix below:

```
1 1 1 1 1 1 1 1 1 1
1 1 1 5 5 5 5 5 5 1
1 1 5 1 1 1 1 1 1 1
1 1 5 1 1 12 1 12 1 1
1 5 1 1 1 1 1 1 5 1
1 5 1 1 1 1 1 1 5 1
1 5 1 1 1 1 1 1 5 1
1 1 5 1 1 1 1 1 1 1
1 1 1 5 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
```

and row = 4, column = 2, and newColor = 33 the resulting matrix would be

```
1 1 1 1 1 1 1 1 1 1
1 1 1 5 5 5 5 5 5 1
1 1 5 33 33 33 33 33 5 1
1 1 5 33 33 33 33 33 5 1
1 1 33 33 33 33 33 33 5 1
1 1 33 33 33 33 33 33 5 1
1 1 33 33 33 33 33 33 5 1
1 1 33 33 33 33 33 33 5 1
1 1 33 33 33 33 33 33 5 1
1 1 33 33 33 33 33 33 5 1
```

Notice the floodFill stopped at the boundary set up by the 5s.
Another example:

If canvas is equal to the matrix below:

```
1 1 1 1 1 1 1 1 1 1
1 1 1 5 5 5 5 5 5 1
1 1 5 1 1 1 1 1 5 1
1 5 1 1 12 1 12 1 5 1
1 5 1 1 1 1 1 1 5 1
1 5 1 1 55 55 55 1 5 1
1 1 5 1 1 1 1 1 5 1
1 1 1 5 1 1 1 1 5 1
1 1 1 5 5 5 5 5 5 1
1 1 1 1 1 1 1 1 1
```

and row = 0, column = 0, and newColor = 3 the resulting matrix would be

```
3 3 3 3 3 3 3 3 3 3
3 3 3 5 5 5 5 5 5 3
3 3 5 1 1 1 1 1 5 3
3 5 1 1 12 1 12 1 5 3
3 5 1 1 1 1 1 1 5 3
3 5 1 1 55 55 55 1 5 3
3 3 5 1 1 1 1 1 5 3
3 3 3 5 1 1 1 1 5 3
3 3 3 5 5 5 5 5 5 3
3 3 3 3 3 3 3 3 3 3
```

Notice the floodFill stopped at the boundary set up by the 5s and the limits or boundaries of the matrix.

Complete method floodFill on the next page.
public void floodfill(int[][] canvas, int row, int col, int newColor) {
/* pre: canvas != null, canvas is a rectangular matrix. (all rows have the same number of columns)
   0 <= row < canvas.length, 0 <= col < canvas[0].length
   post: flood fill performed as explained above.
*/
3. Implementing Simple Data Structures (25 points) Consider an ArrayList as discussed in class. Implement an add method where the user specifies the position in the list to add the new element. You must also answer the three questions below about the Big O of this method as you have implemented it.

public class ArrayList
{
    // storage container for items. Items are
    // in the first iMySize elements.
    private Object[] myCon;

    // number of items in the List. May be less than or equal to
    // myItems.length
    private int iMySize;

    // other methods and private data members not shown

    // you may use the following methods in writing your method
    // but no others

    // pre: newCapacity > size()
    // post: capacity of ArrayList >= newCapacity
    public void ensureCapacity(int newCapacity)
    {
        // pre: none.
        // post: returns number of items in list
        public int size()
    }

    // complete the following method
    // pre: 0 <= pos <= size()
    // post: size() = old size() + 1, get(pos) = item
    //       all old items after this one shifted down by 1
    public void add(Object item, int pos)
    {
        // complete this method on the next page

What is the Best Case Big O of this method? __________

What is the Worst Case Big O of this method? __________

What is the Average Case Big O of this method? _______
// pre: 0 <= pos <= size()
// post: size() = old size() + 1, get(pos) = item
//       all old items after this one shifted down by 1
public void add(Object item, int pos)
{

4. Creating Classes (20 Points) Internet Protocol addresses (IP addresses) are used to identify all the computers connected to the internet. You may have see IP addresses before. For example the IP address for one of Yahoo's five web servers is 66.218.71.63. The IP address is broken up into 4 octets (66 is the first octet, 63 is the last octet.). Each octet in the IP address can have values between 0 and 255 inclusive. Each IP address also has a list of domain names. These are the more human readable strings you may be familiar with such as "www.yahoo.com" or "yahoo.com". When you type in the domain name in your browser a lookup is performed to get the IP address associated with the domain name. You will write part of a class that models an IP address and provides some behavior.

Write a class IPAddress. Each IP Address consists of 4 octets, each octet may take values between 0 and 255 inclusive. Each IPAddress also has a list of associated domain names such as www.yahoo.com. You may assume no IPAddress may have more than 25 domain names associated with it. Complete the IPAddress class with the following methods. (Note this is not a complete class, you are only implementing part of it. Do not assume any other method exist):

1. list the private data members you will use. Native arrays are okay, but you may not use ArrayList.

2. a method that takes in a value between 1 and 4 and returns the associated octet as a byte (one of the primitive integer data types in Java.)

3. a method to add a domain name to the list of domain names.

4. a method that returns the number of domain names this IPAddress has.

5. Override the equals method. Two IPAddresses (the calling object and the other object) are equal. Two IPAddresses are equal if they have the exact same octets in the same order.

6. a method to set the IP address. The method has one parameter, an array of booleans. There are 32 booleans in the array used to represent 32 bits. false represents a 0 and true represents a 1. The first 8 elements of the array make up the first octet, the second eight elements the second octet and so forth. 8 falses would represent the octet 0 and 8 trues would represent 255.

More examples:
false false false false false false false true represents binary number 00000001 or the octet 1
false false false false true true false true represents the binary number 00001101 or the octet 13

Note, we could have made the parameter a single int (which is a 32 bit signed integer) but then I would have had to explain 2's compliment and representation of negative numbers or bit operators and I just don't want you to deal with that at this moment.
Complete the class below. You must only do the parts specified on the previous page. Do not assume any other methods exist although you may add helper methods if you wish.

```java
public class IPAddress {
```