CS112 Data Structures - Midterm 1 - Spring 2022

Name: ______________________________________ NetID: ___________________________

- **WRITE your name and NetID on EVERY page.**
- **DO NOT REMOVE** THE STAPLE IN YOUR EXAM.
- **DO NOT BEGIN** UNTIL INSTRUCTED TO DO SO.
- WRITE NEATLY AND CLEARLY. If we cannot read your handwriting, you will not receive credit. Please plan your space usage. No additional paper will be given.
- This exam is worth 150 points.

## Problem 1 – Arrays (30 points)

Given the code segment below.

```
for ( int i = 0; i < n - 1; i++ ) {

    int min = i;

    for ( int j = i + 1; j < n; j++ ) {

        if ( a[j] < a[min] ) {
            min = j;
        }
    }

    int temp = a[i];
    a[i] = a[min];
    a[min] = temp;
}
```

(a) **(10 points)** How many times is the `if` statement in the inner `for` loop executed? Give your answer as a function of n with a succinct explanation.

(b) **(10 points)** What is the maximum number of array accesses (reads and writes) for the entire code segment as a function of n? Justify your answer.

(c) **(5 points)** Write the tilde notation for the function in (b).

(d) **(5 points)** Write the Big-O notation for the function in (b).

Name: ______________________________________ NetID: ___________________________

## Problem 2 – Union-Find (30 points)

A client program is using the union-find API to solve a dynamic connectivity problem in networking. Assume that there are 10 sites identified as: 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9 in the network. Answer the following questions.

(a) **(10 points)** If the union-find API is implemented with the quick-find discussed in class, show the content of the `id[]` array after adding the following edges in sequence: 0-7, 1-5, 2-8, 7-1, 8-3

(b) **(10 points)** If the API is implemented with quick-union discussed in class, show the content of the `parent[]` array after adding the following edges in sequence: 0-7, 1-5, 2-8, 7-1, 5-2

(c) **(10 points)** If the API is implemented with weighted-quick-union discussed (union by size) in class, show the content of the `parent[]` and `size[]` array after adding the same edges from part (b).

## Problem 3 – Stacks and Queues (30 points)

(a) **(8 points)** What is the output of the following code segment?

```
Stack<Integer> stack = new Stack<Integer>();
int item1 = 1;
int item2 = 0;
int item3 = 4;

stack.push( item2 );
stack.push( item1 );
stack.push( item1 + item3 );

item2 = stack.pop();

stack.pop();
stack.push( item3 * item3 );
stack.push( item2 );
stack.push( 3 );

item1 = stack.pop();

stack.pop();
System.out.println( item1 + " " + item2 + " " + item3 );

while ( !stack.empty() ) {
    System.out.println( stack.pop() );
}
```

Answer:

(b) **(8 points)** What is the output of the following code segment?

```
 Queue<Character> queue = new Queue<Character>();
Stack<Character> stack = new Stack<Character>();

String s = "stack and queue";

char [] charArray = s.toCharArray();

for ( int i = 0; i < charArray.length; i++ ) {
    stack.push( charArray[i] );
}

while ( !stack.isEmpty() ) {
    queue.enqueue(stack.pop());
}

while ( !queue.isEmpty() ) {
    System.out.print( queue.dequeue() );
}
```

Answer:

Name: ________________________________________ NetID: ____________________________

(c) Answer questions based in the code segment below. Note that randomBoolean() randomly returns true or false.

```
Stack<Integer> stack = new Stack<>();

for ( int count = 1; count <= 5; count++ ) {

    if ( randomBoolean() ) {
        System.out.print( count );
    } else {
        stack.push( count );
    }
}

while ( !stack.isEmpty() ) {
    System.out.print( stack.pop() );
}
```

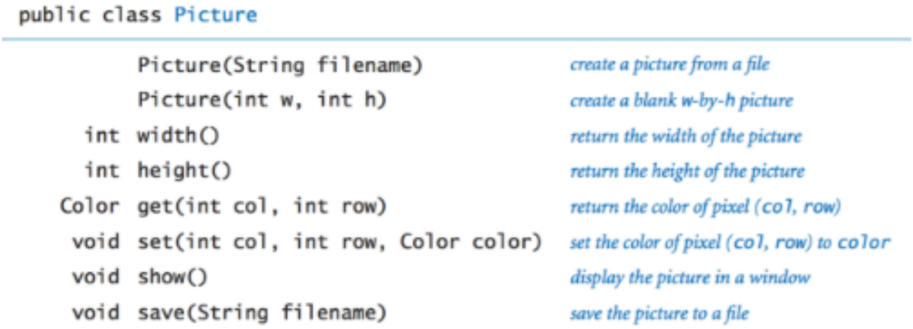
1. **(7 points)** Is the output 13524 possible? Explain.

2. **(7 points)** Is the output 13542 possible? Explain.

Name: ____________________________________ NetID: _________________________

## Problem 4 – Linked Lists and Arrays (30 points)

On ArtCollage you worked with the Picture class which follow the digital image abstraction. The Picture is a 2D array of Color values, see operations below.



```
public class Picture

             Picture(String filename)                create a picture from a file
             Picture(int w, int h)                   create a blank w-by-h picture
         int width()                                 return the width of the picture
         int height()                                return the height of the picture
       Color get(int col, int row)                   return the color of pixel (col, row)
        void set(int col, int row, Color color)      set the color of pixel (col, row) to color
        void show()                                  display the picture in a window
        void save(String filename)                   save the picture to a file
```

The ImageProcessing class below also uses the Picture class, it contains two instance variables:

- ***image***: a reference to a Picture.
- ***uniqueColorList***: a reference to the front node of a linked list that stores all the unique colors from the image.

The class also contains the methods:

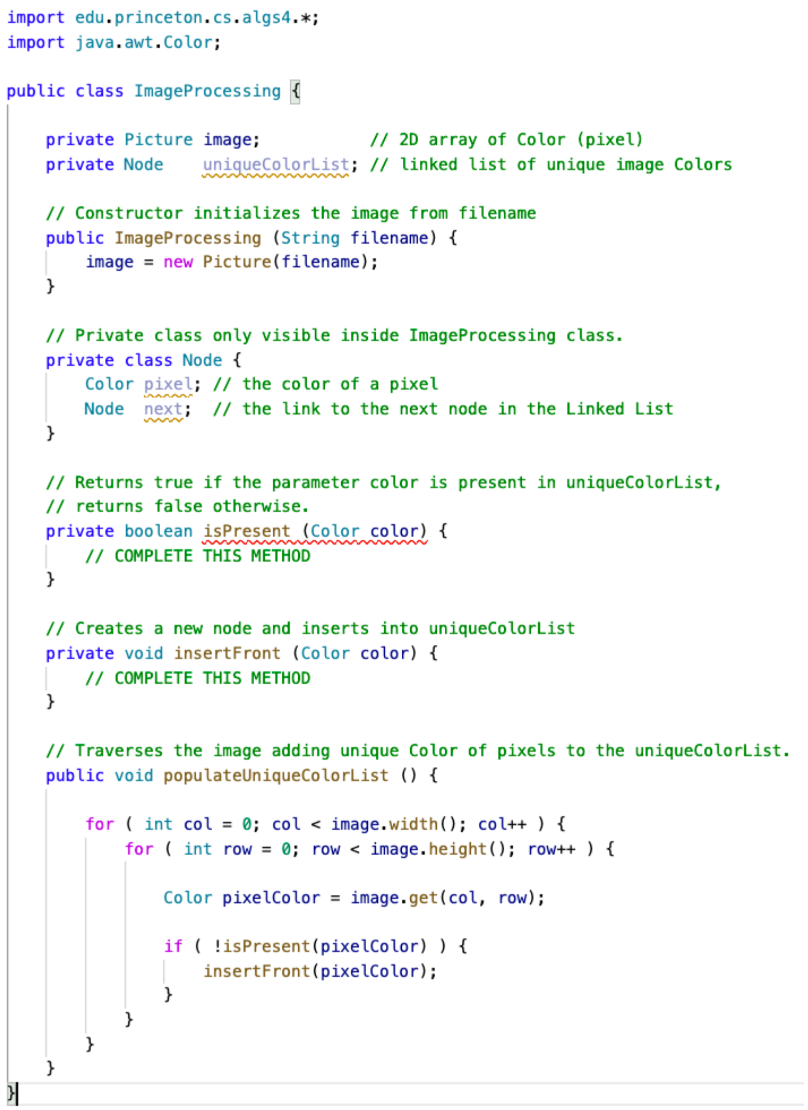
- ***populateUniqueColorList***: that inserts in the *uniqueColorList* all pixel colors that are unique. The list will NOT contain duplicate colors.
- ***isPresent(Color color)***: returns true if the parameter *color* is present in the *uniqueColorList*.
- ***insertFront(Color color)***: creates a new linked list node where *pixel* refers to the color of the pixel, and inserts the newly created node at the front of the *uniqueColorList*.

a) **(15 points)** Implement the ***isPresent*** method.

```
private boolean isPresent (Color color) {
```

b) **(15 points)** Implement the ***insertFront*** method.

```
private void insertFront (Color color) {
```



```
import edu.princeton.cs.algs4.*;
import java.awt.Color;

public class ImageProcessing {

    private Picture image;          // 2D array of Color (pixel)
    private Node    uniqueColorList; // linked list of unique image Colors

    // Constructor initializes the image from filename
    public ImageProcessing (String filename) {
        image = new Picture(filename);
    }

    // Private class only visible inside ImageProcessing class.
    private class Node {
        Color pixel; // the color of a pixel
        Node  next;  // the link to the next node in the Linked List
    }

    // Returns true if the parameter color is present in uniqueColorList,
    // returns false otherwise.
    private boolean isPresent (Color color) {
        // COMPLETE THIS METHOD
    }

    // Creates a new node and inserts into uniqueColorList
    private void insertFront (Color color) {
        // COMPLETE THIS METHOD
    }

    // Traverses the image adding unique Color of pixels to the uniqueColorList.
    public void populateUniqueColorList () {

        for ( int col = 0; col < image.width(); col++ ) {
            for ( int row = 0; row < image.height(); row++ ) {

                Color pixelColor = image.get(col, row);

                if ( !isPresent(pixelColor) ) {
                    insertFront(pixelColor);
                }
            }
        }
    }
}
```

## Problem 5 – Binary Search Tree (BST) (30 points)

(a) **(7 points)** After inserting 18, 51, 37, 11, 46, 25, and 20 into an empty BST in that order, what would be the worst case number of comparisons (`compareTo` calls) for a successful search?

(b) **(8 points)** If we perform searches for 1, 2, 6, 7, 9, 11, 20 in the following BST, what would be the average number of comparisons (`compareTo` calls), regardless of whether the search ends in success or failure? Give the reasoning.

```
   7
  / \
 4   9
  \
   6
```

(c) **(8 points)** In the worst case scenario, what would be the time complexity to delete the node containing the largest value in a BST? Give the reasoning.

(d) **(7 points)** Given the following numbers to insert into an empty BST: 2, 7, 8, 10, 15, 20. What insertion order would yield the tree with the least height?

A. 15, 2, 20, 8, 7, 10
B. 8, 20, 7, 2, 15, 10
C. 7, 2, 10, 8, 15, 20
D. 10, 7, 15, 20, 2, 8