CompSci-61B, Data Structures – Midterm Exam

Your Name: ______________________________

Your 8-digit Student ID: ______________________________

This is a midterm test for mastery of the material covered in our labs, lectures, and readings from *introduction to Java* through *sorting*. It is worth 100 points. Circle the correct answer for the multiple-choice questions, and write in the space provided for the essay questions. This exam is *open book* and *open note*. This exam consists of 28 questions: 16 worth 2 points each, 10 worth 4 points each, an 8-pointer, and a 20-pointer. Partial credit is possible.

1. If the integer **`a`** has a value of 11, and is declared as an **`int`**, what is *the exact numeric result* of this Java expression: **`( a / 3 )`** ? **[2 points]**

   3

2. Write the *logical opposite* of this logical expression: **`score >= 0 && score <= 100`** **[2 points]**

   !(score >= 0 && score <= 100)

3. Comment on the following program: *(circle one)* **[2 points]**

```
public class MidtermExam
{
  public static void main(String[] argv)
  {
    int a = 89;
    int b = 99;
    System.out.print((a + b) / 2);
  }
}
```

(a) will *not* compile because **`try/catch`** is missing

[✓ (b) will compile and run]

(c) will *not* compile because division is not allowed for integers

(d) will *not* compile because there is a missing **`import`**

4. For a Java array of references to objects of class **X** to be traversed with a for-each loop, what must be true about class **X**? **[2 points]**

   Class x must implement iterable

5. Does the programmer have to write one or more constructor methods for *every class* that has private data members? Explain. **[2 points]**

   (a) yes, one or more contructors must be written… [✓ (b) no, constructors do not always have to be written…]

   explain: If a class does not have a constructor, primitives will be initialized to 0 or false and object instances will be initialized to null by the default constructor.

6. What gets printed by the following code block?
   Look carefully – it's not as easy as it may first appear! *(Circle one)* **[2 points]**

```
// count to 3
int i;
for (i = 0; i < 3; i++);
  System.out.print("" + i + ' ');
System.out.println();
```

(a) **1 2 3** (b) **0 1 2** (c) **0 1 2 3** [✓ (d) **3**] (e) nothing

7. The following code compiles and runs. But it does not behave as its programmer intended. Explain the *logic* error. Hint: the problem is *not* with the console input programming: **[4 points]**

Explain: It will always print out "Wow – that's pretty good!" regardless of score.

```java
import java.io.*;

public class MidtermExam
{
  public static void main(String[] argv) throws Exception
  {
    BufferedReader cin;
    cin = new BufferedReader(new InputStreamReader(System.in));

    int score;
    System.out.print("Enter your CS61B midterm exam score: ");
    score = new Double(cin.readLine()).intValue();

    if (score >= 80 || score <= 100)
      System.out.println("Wow – that's pretty good!");
    else
      System.out.print("Hmm... study harder!");
  }
}
```

8. The following code will not compile. Explain the coding error that prevents the code from compiling. Hint: the problem is *not* with the console input programming: **[4 points]**

Explain: The first if statement does not have brackets, and is more than one line, so the else if statement does not have a related if statement

```java
import java.io.*;

public class MidtermExam
{
  public static void main(String[] argv) throws Exception
  {
    BufferedReader cin;
    cin = new BufferedReader(new InputStreamReader(System.in));

    int score;
    System.out.print("Enter your CS61B midterm exam score: ");
    score = new Double(cin.readLine()).intValue();

    if (score == 100)
      System.out.println("Wow – that's perfect!");
      System.out.println("You get an A!");
    else if (score >= 80)
      System.out.println("Pretty good!");
    else
      System.out.println("Hmm... maybe you need to study more!");
  }
}
```

9. The following code will not compile! Explain what I am doing wrong. Hint: the problem is *not* with the console input programming: **[4 points]**

Explain: sum is not initialized before reaching line 18

```
 1 import java.io.*;
 2
 3 public class MidtermExam
 4 {
 5   public static void main(String[] argv) throws Exception
 6   {
 7     BufferedReader cin;
 8     cin = new BufferedReader(new InputStreamReader(System.in));
 9
10     int score;
11     int sum;
12     while (true)
13     {
14       System.out.println("Enter a CS61B midterm exam score [-1 to exit]: ");
15       score = new Double(cin.readLine()).intValue();
16
17       if (score < 0) break;
18       sum = sum + score;
19     }
20     System.out.println("The total of all scores is: " + sum);
21   }
22 }
```

Write the big oh of these operations in questions 10-12:

10. **quick sort, randomly-ordered array**? *fill in the blank:* O(n log n) **[2 pts]**

11. **insertion sort of an already-sorted array**? *fill in the blank:* O(n) **[2 pts]**

12. **selection sort of a simple, singly-linked list, without a “current node”**? *fill in the blank:* O(n^2) **[2 pts]**

13. A certain search algorithm scales by O(1) when applied to a data structure containing n values. In a timing test with 1000 values, the search took 5 milliseconds to complete. How many milliseconds would you expect another timing test to take with 2000 values, or 4000 values? *(Circle one each)* **[2 points]**

with 2000 values : [✓ (a) 5 ms] (b) 6 ms (c) 10 ms (d) 20 ms

with 4000 values: [✓ (a) 5 ms] (b) 7 ms (c) 20 ms (d) 80 ms

14. A certain search algorithm scales by O(log n) when applied to a data structure containing n values. In a timing test with 256 values, the search took 8 milliseconds to complete. How many milliseconds would you expect another timing test to take with 512 values, or 1,024 values? *(Circle one each)* **[4 points]**

with $\overset{512}{256}$ values : (a) 8 ms [✓ (b) 9 ms] (c) 16 ms (d) 32 ms

with 1,024 values: (a) 8 ms [✓ (b) 10 ms] (c) 32 ms (d) 256 ms

15. A certain sort algorithm scales by O(n^2) when applied to a data structure containing n values. In a timing test with 100 values, the search took 10 milliseconds to complete. How many milliseconds would you expect another timing test to take with 200 values, or 400 values? *(Circle one each)* **[2 points]**

with 200 values: (a) 10 ms (b) 20 ms (c) 30 ms [✓ (d) 40 ms]

with 400 values: (a) 20 ms (b) 40 ms (c) 80 ms [✓ (d) 160 ms]

16. A certain sorting algorithm scales by O(n log n) when applied to a data structure containing n values. In a timing test with 256 values, the search took about 20 milliseconds to complete. How many milliseconds would you expect another timing test to take with 512 values, or 1,024 values? *(Circle one each)* **[4 points]**

512
with 256 values: (a) about 37 ms [✓ (b) about 46 ms] (c) about 80 ms

with 1,024 values: (a) about 46 ms (b) about 80 ms [✓ (c) about 102 ms]

17. How many times will the prompt to enter an exam score appear in the following Java program? Assume that the user enters only valid numeric values – no CTRL-C to terminate the loop early. *(circle one)* **[4 points]**

```
import java.io.*;

public class MidtermExam
{
  public static void main(String[] argv) throws Exception
  {
    BufferedReader cin;
    cin = new BufferedReader(new InputStreamReader(System.in));

    int count = 0;
    while (true)
    {
      if (count == 10) break;

      int score;
      System.out.println("Enter a CS61B midterm exam score: ");
       score = new Double(cin.readLine()).intValue();
       if (score >= 40)
         System.out.println("Very good!");
       count = count + 1;
    }
  }
}
```

(a) nine [✓ (b) ten] (c) eleven (d) it depends on what numbers that the user enters

18. How many objects are created by executing the following statement. *(circle one)* **[2 points]**

```
String[] a = new String[10];
```

(a) none [✓ (b) one] (c) ten (d) eleven

19. *Markup* the following traversal of a doubly-linked list so that it goes backwards from the tail instead of forwards from the head: **[2 points]**

```
                 tail                         previous
for (Node p = head; p != null; p = p.next)
   System.out.println("Value: " + p.value);
```

20. What does the keyword **`static`** mean in this data declaration inside the public class named `X`:

`private static double PI = 3.14159;` ? **[2 points]**

It means that PI is a class variable

21. What does the keyword **`final`** mean in this data declaration inside the public class named `X`:

`private final double PI = 3.14159;` ? **[2 points]**

it means PI is a constant (cannot be changed once initialized)

22. This Java statement : **`if (a >= b)`** does *not* do a lexicographical comparison of the two String references, **`a`** and **`b`**. Rewrite this so that it *does* do a case-dependent, lexicographical comparison of these two: **[2 points]**

`if(a.compareTo(b) >= 0)`

23. Declare private data members to add a "current node" reference to a singly-linked list: **[4 points]**

```
public class MySinglyLinkedList<T> implements MyList<T>
{
   ... // the private inner Node class with a value and a next reference only – no prev reference

  private Node head;
  private int nValues;                                   Fill in this box
```

```
private Node currNode;
```

```
   ... // the rest of the class...
}
```

24. In our `MyArrayList`, every single insertion is fast, O(1), except for the ones that expand the array since they need to copy the entire contents of the old array to a new one. Min observed this and claims that we can make `MyArrayList.add` more efficient by squaring the size of the array when expanding instead of doubling. His rationale is that we would not have to expand the array nearly as often this way. Describe to him why squaring the size offers no faster running time over just doubling the size. **[4 points]**

Explain using big oh reasoning: Squaring the size offers no faster running time since declaring the array is O(n).

25. Write a **`getNodeAt`** method for the singly-linked list from the previous question, with assertions to check for programming logic errors: **[4 points]**

```
public class MySinglyLinkedList<T> implements MyList<T>
{
   ... // the private inner Node class, private data members, accessors and mutators…
  private Node getNodeAt(int index)  // traverse from head or current node, whichever's faster to do
  {
```

```
  int node = head;
  for(int i = 0; i < index && node.next != null; i++) node = node.next;
  if(node.next == null) return null;
  return node;
```

```
  }
}
```

26. Assume that `Number` is a Java interface that `Integer`, `Double`, `Long`, and other classes implement, and that `Number` has a `isPositive` accessor method that tests for whether the value is positive. We would like to declare a method that scans through a list of `Number`s and return the number of positive elements. We would like our method to accept as many input types as possible. **[4 points]**

First, explain why the following declaration won't work for lists declared as `List<Integer>`, `List<Double>`, `List<Long>`, etc.:

`public static int countPositive(List<Number> ls) { ... }`

You cannot instantiate a generic list with an interface

Now, fill in the correct declaration: *(fill in the blank)*

`public static int countPositive(List<` E implements Number **`> ls) { ... }`**

27. Predict the output of the following programs. Some of the lines may generate compiler or runtime error; if so, indicate it on your answer and treat the rest of the program as if that line did not exist. **[8 points]**

```
public class AString
{
  public String body;
  public AString(String text) { body = text; }
  public String toString() { return body;}

  public static void main(String[] argv)
  {
    AString a = new AString("Blessed are the poor in spirit,");
    AString b = new WeirdString(a.toString());
    AString c = new WeirdString(new AString(a.toString()));
    System.out.println(b == a); // output line 1
    System.out.println(b.equals(a)); // output line 2
    System.out.println(b.toString() == a.toString()); // output line 3
    System.out.println(b.toString().equals(a.toString())); // output line 4
    System.out.println(c.toString() == b.toString()); // output line 5
    System.out.println(c.toString().equals(b.toString())); // output line 6
    a.body = "For theirs is the kingdom of heaven";
    System.out.println(b); // output line 7
    System.out.println(c); // output line 8
  }
}

class WeirdString extends AString
{
  public String body;
  public WeirdString(String text)
  {
    super(text);
    body = text;
  }

  public String toString() { return body;}
}
```

Write the output you would expect for the 8 lines of output in this program, per the numbering in ***comments*** above.

1. false
2. false
3. false
4. true
5. false
6. true
7. “Blessed are the poor in spirit”
8. “Blessed are the poor in spirit”

28. Write code in the boxes on the next three pages. **[20 points]**

A matrix is a rectangular table of numbers. For example, a 3x3 matrix might look like

```
1 0 0
0 1 0
0 0 1
```

You can add two matrices of the same dimension by simply adding together their corresponding elements with the same row and column index. For this problem, we will build a simple integer matrix data structure based only on lists that you built in lab. You may ONLY use variables of type **MyList** and **`int`**/**`Integer`**, e.g., **`MyList<Integer> a =`**... would be okay while `Integer[] a =`... and `MyArrayList<Integer> a = ...` would not be acceptable. Fill in the following program so that it conforms to the comments, and with proper index validation and with assertions to check for programming logic errors. Do *not* deal with packaging and imports. The interface MyList is provided for ease of reference.

```
public interface MyList<T> extends Iterable
{
  public boolean add(T value);
  public boolean add(int index, T value);
  public T remove(int index);
  public void clear();
  public boolean replace(int index, T value);
  public T getEntry(int index);
  public boolean contains(T value);
  public int size();
  public boolean isEmpty();
  public boolean isFull();
}

public interface IntMatrix // Interface for a Matrix of Integers
{
  // Mutator: sets the specified element of the matrix to new_val
  // returns false if the modification failed. Assume 0 indexing
  public boolean set(int row, int col, Integer new_val);

  // Accessor: returns the specified element of the matrix. Assume 0 indexing.
  public Integer get(int row, int col);

  public int getRowNum();//Accessor: returns #of rows in the matrix
  public int getColNum();//Accessor: returns #of columns in the matrix
```

continued on next page

```
  // Mutator: add this matrix to addee and store the result in this matrix. Returns false if the addition fails
  public boolean add(IntMatrix addee);

  // Accessor: print entire content of matrix in any format so long as user sees when one row begins and another ends.
  public void print();
}

// Our implementation of IntMatrix of a matrix with "n" rows x "m" columns
public class ListIntMatrix implements IntMatrix
{
  // Declare your private data members here.
```

Fill in this box

```
    private Integer[][] matrix;
```

```
  // Default constructor creates a 1x1 matrix
  public ListIntMatrix()
  {
    this(1,1);
  }

  // Creates a matrix of size row_num x col_num. The matrix is
  // initially filled completely with 0's
  public ListIntMatrix(int row_num, int col_num) // constructor
  {
```

Fill in this box

```
     matrix = new Integer[row_num][col_num];
```

```
  }
```

continued on next page

// write all additional methods here – you may include private methods *Fill in this box*

```
public Integer get(int row, int col) {
    return matrix[row][col];
}

public int getRowNum() {
    return matrix.length;
}

public int getColNum() {
    return matrix[0].length;
}

public boolean add(IntMatrix addee) {
    if(getRowNum() != addee.getRowNum() || getColNum() != addee.getColNum()) return false;
    for(int r = 0; r < getRowNum(); r++)
        for(int c = 0; c < getRowNum(); c++)
            matrix[r][c] += addee[r][c];
    return true;
}

public boolean set(int row, int col, Intger newVal) {
     if(row >= getRowNum() || col >= getColNum()) return false;
     matrix[row][col] = newVal;
     return true;
}

public void print() {
     for(int r = 0; r < getRowNum(); r++) {
          for(int c = 0; c < getRowNum(); c++) System.out.print(matrix[r][c] + ″ ″);
          System.out.println(“”);
     }
}
```