Name: __________________________

New York University
Introduction to Computer Science
Midterm2B Sample Problems – Solution Key

Instructions:

KEEP TEST BOOKLET CLOSED UNTIL YOU ARE INSTRUCTED TO BEGIN.

This exam is double sided (front and back)!

No calculators, notes, textbooks, or any other aids are allowed except writing utensils (pens, pencils, crayons, erasers, etc.) or other aids provided to you by the instructor. If you need extra scratch paper, please pick it up from the front of the class.

You should also be provided with an appendix that provides helpful documentation.

All answers must be submitted on (or attached to) this exam sheet. All answers must be clearly legible.

Commenting code: Comments aren’t required, but if you provide good comments, they can earn you partial credit if your code isn’t correct.
Example True or False (10 points):

Instructions: Circle either True or False based on the validity of the statement.

1. In object oriented programming, the superclass inherits properties from the subclass.
   true  false

[There are usually 5-10 questions of this type]
Example Multiple Choice (10 points):
Instructions: Circle the letter of the best answer.

1. The data for which of these data types is stored in the Heap memory (select all that apply):
   A. boolean
   B. int
   C. double
   D. String
   E. int[]

[There are usually 5-10 questions of this type]
Example Short Answer (10 points):
Instructions: In your own words, answer the questions as best as possible in one or two sentences.

1. What is the difference between a Class and an Object.

   A class provides the blueprint for creating new object instances and what data and methods each object will have. Classes can contain static fields and static methods that are shared between all objects. Objects store data associated with that particular object and each objects member fields are unique to that object.

   [There are usually 5-10 questions of this type]
Example Entomology - Study of Bugs (10 points):

Instructions: Find 3 bugs in the following program (there are more than three):

1) Specify the line number of the bug (Note: there can be more than one bug per line)
2) Check the box next to the type of error; either “Logic” for logic errors or “Other” for (syntax, type, etc.).
3) Write a short explanation or fix for the error.

<table>
<thead>
<tr>
<th>Bug</th>
<th>Line #</th>
<th>Type of Error</th>
<th>Explanation or Fix</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>Other (unresolved symbols)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>// syntax error</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>18</td>
<td>// syntax error</td>
<td></td>
</tr>
<tr>
<td></td>
<td>26</td>
<td>// syntax error</td>
<td></td>
</tr>
<tr>
<td></td>
<td>29</td>
<td>// syntax error</td>
<td></td>
</tr>
<tr>
<td></td>
<td>33</td>
<td>// syntax error</td>
<td></td>
</tr>
<tr>
<td></td>
<td>34</td>
<td>// syntax error</td>
<td></td>
</tr>
<tr>
<td></td>
<td>35</td>
<td>// syntax error</td>
<td></td>
</tr>
<tr>
<td></td>
<td>36</td>
<td>// syntax error</td>
<td></td>
</tr>
</tbody>
</table>
Example What is the Output (10 points):

Instructions: What is the output if the following code is run?

```java
public class InsertionSort {
    public static void main(String[] args) {
        int[] ints = { 9, 5, 7, 1, 3 };

        int indexUnsorted;
        int itemToCompare;
        int indexSortedSorted;

        // move unsorted items their proper spot in the sorted part of the array
        for (indexUnsorted = 1; indexUnsorted < ints.length; indexUnsorted++) { // skip first item
            itemToCompare = ints[indexUnsorted];
            // move value to proper position in sorted list
            for (indexSortedSorted = indexUnsorted - 1; (indexSortedSorted >= 0) && (itemToCompare <
                    ints[indexSortedSorted]); indexSortedSorted--) {
                ints[indexSortedSorted + 1] = ints[indexSortedSorted]; // move smaller values down
            }
            ints[indexSortedSorted + 1] = itemToCompare; // Put the value in its proper location

            // After every item is moved, print out the new array
            for (int i : ints)
                System.out.print(i + " ");
            System.out.println();
        }
    }

    /**
     * Takes and array and swaps values from <code>index1</code> and <code>index2</code>
     *
     * @param ints - array of values
     * @param index1 - one of the indexes to swap the contents of
     * @param index2 - the second index to swap the contents of
     */
    public static void swap(int[] ints, int index1, int index2) {
        int temp = ints[index1];
        ints[index1] = ints[index2];
        ints[index2] = temp;
    }
}
```

Answer:

```
5 9 7 1 3
5 7 9 1 3
1 5 7 9 3
1 3 5 7 9
```
Example Comment the Code (10 points):

Instructions: The following code was written, but the documentation was left out. Analyze the code to determine what each function does and then what the program as a whole does. Fill in the missing comments. Be as specific as possible. Remember that Javadocs should explain WHAT the code does, and inline comments are generally best if they explain why the code is doing what it does.

```java
/**
 * Generates an array of random numbers between <code>min</code> and <code>max</code> inclusively.
 * @param size size of the array to create
 * @param min min minimum value in the array
 * @param max max maximum value in the array
 * @return a list of random numbers
 */
static int[] dostuff(int size, int min, int max)
{
    Random rand = new Random();

    // Create an array of the size passed in
    int[] ints = new int[size];

    // For each value in the array assign it a random value
    for (int i = 0; i < ints.length; i++) {
        int nextRandom = rand.nextInt(max-min + 1);
    }

    // put it back in the range from min-max
    nextRandom += min;

    // place that random number in the array
    ints[i] = nextRandom;
}
return ints;
}

/**
 * Get the sum of all values in an array <code>ints</code>.
 * @param ints array of integers to sum
 * @return the sum of all values in the array
 */
static int doSomething(int[] ints) {
    int sum = 0;

    // Add all values in the array
    for (int i = 0; i < ints.length; i++) {
        sum += ints[i];
    }

    // return the sum of all values
    return sum;
}
Example Fill in the Code (10 points):

Instructions: Fill in the missing code.

```java
public class Player {
    String name;
    int wins, losses; // Number of wins and loses this player has had
    int health = 100; // Percentage from 0-100, if 0, player is dead

    Player() {
        name = "Player";
    }

    /**
     * Copy constructor
     *
     * @param player to make a copy of
     */
    public Player(Player player) {
        this.name = player.name;
        this.wins = player.wins;
        this.losses = player.losses;
        this.health = player.health;
    }

    /**
     * @return a String representation of this player
     */
    public String toString() {
        return "Player '" + name + "' wins=" + wins + " losses=" + losses + " health=" + health;
    }

    /**
     * Determines if this player is equivalent to another player
     *
     * @param player to compare this player to
     * @return true if this player and <code>player</code> have all equivalent data, otherwise false
     */
    public boolean equals(Player player) {
        if (!this.name.equals(player.name)) return false;
        if (this.wins != player.wins) return false;
        if (this.losses != player.losses) return false;
        if (this.health != player.health) return false;
        return true;
    }

    // Rest of code cut for brevity
}
```
Example Write the Code:

In java code, write a class that has the following data and methods associated with it:

Ball:
---data-----------------
- xPosition: int
- yPosition: int
- xSpeed: int
- ySpeed: int
- size: int
---methods------------
+ Ball()
+ Ball(xPosition: int, yPosition: int, xSpeed: int, ySpeed: int, size: int)
+ setPosition()
+ reverseSpeedY()
+ reverseSpeedX()

```java
class Ball {
    protected int xPos;
    protected int yPos;
    protected int xSpeed;
    protected int ySpeed;
    protected int size;

    public Ball() {
    }

    public Ball(int xPosition, int yPosition, int xSpeed, int ySpeed, int size) {
        this.xPosition = xPosition;
        this.yPosition = yPosition;
        this.xSpeed = xSpeed;
        this.ySpeed = ySpeed;
        this.size = size;
    }

    public void setPosition(int x, int y) {
        xPos = x;
        yPos = y;
    }

    public void reverseY() {
        ySpeed = -ySpeed;
    }

    public void reverseX() {
        xSpeed = -xSpeed;
    }
}
```
Example Write the Code:

In Java code, write the following method that implements Selection Sort:

```java
public static void selectionSort(int[] array) {
    int currentMin; // Current minimum found in unsorted portion of the list
    // advance the position through the entire array
    // (can do j < length-1 because single element is also min element)
    for (int j = 0; j < array.length - 1; j++) {
        // find the min element in the unsorted array[j .. n-1]
        // assume the min is the first element in the unsorted values
        currentMin = j;

        // test against elements after j to find the minimum
        for (int i = j + 1; i < array.length; i++) {
            // if this element is smaller, then it is the new minimum
            if (array[i] < array[currentMin]) {
                // found new minimum; remember its index
                currentMin = i;
            }
        }

        // currentMin is the index of the minimum element.
        // Swap it with the current position to update the sorted list
        if (currentMin != j) {
            swap(array, j, currentMin);
        }

        // Print out the results from this iteration of selection sort
        for (int i : array)
            System.out.print(i + " ");
        System.out.println();
    }
}
```

```java
/**
 * Takes an array and swaps values from <code>index1</code> and <code>index2</code>
 * @param ints - array of values
 * @param index1 - one of the indexes to swap the contents of
 * @param index2 - the second index to swap the contents of
 */
public static void swap(int[] ints, int index1, int index2) {
    int temp = ints[index1];
    ints[index1] = ints[index2];
    ints[index2] = temp;
}
```