CIS 110 — Introduction to Computer Programming
Summer 2016 — Final

Name: ______________________________

Recitation # (e.g., 201): ______________________________

Pennkey (e.g., paulmcb): ______________________________

My signature below certifies that I have complied with the University of Pennsylvania's Code of Academic Integrity in completing this examination.

______________________________ Signature

______________________________ Date

Instructions:

- **Do not open this exam until told by the proctor**. You will have exactly 120 minutes to finish it.
- **Make sure your phone is turned OFF (not to vibrate!) before the exam starts.**
- Food, gum, and drink are strictly forbidden.
- **You may not use your phone or open your bag for any reason**, including to retrieve or put away pens or pencils, until you have left the exam room.
- This exam is *closed-book, closed-notes, and closed-computational devices*.
- If you get stuck on a problem, it may be to your benefit to move on to another question and come back later.
- All code must be written out in proper java format, including all curly braces and semicolons.
- **Do not separate the pages.** If a page becomes loose, re-attach it with the provided staplers.
- Staple all scratch paper to your exam. Do not take any sheets of paper with you.
- If you require extra paper, please use the backs of the exam pages or the extra pages provided at the end of the exam. **Clearly indicate on the question page where the graders can find the remainder of your work (e.g., "back of page" or "on extra sheet").**
- Use a pencil, or blue or black pen to complete the exam.
- If you have any questions, raise your hand and a proctor will come to answer them.
- When you turn in your exam, you may be required to show ID. **If you forgot to bring your ID, talk to an exam proctor immediately.**
- We wish you the best of luck.

Scores: [For instructor use only]

Question 0		1 pt
Question 1		15 pts
Question 2		10 pts
Question 3		8 pts
Question 4		10 pts
Question 5		5 pts
Question 6		4 pts
Question 7		13 pts
Question 8		9 pts
Total:		**75 pts**

0) (1 point) The Easy One:

- Check that your exam has all 21 pages (excluding the cover sheet).
- Write your name, recitation number, and PennKey (username) on the front of the exam.
- Sign the certification that you comply with the Penn Academic Integrity Code.

1.) RECURSION (15 pts total)

1.1) Binary Search

a) **(4 points)** Consider function `isSorted(int[] array)` below that iterates through an array and determines if the array is sorted in ACSENDING order. This should return true if the non-empty array is sorted and false otherwise. Fill in the blanks of this function.

```
public static boolean isSorted(int[] array){

    for(int i= 0; i < array.length -1; i++){

      if(array[i] > array[i+1]){
        return false;
      }
    }
    return true;
  }
```

.

b) **(1 point)** What would `isSorted(int[] array)` if `array` were empty? `true`

c) **(5 points)** We will write a function `binarySearch(int[] array, int x)` . This function should return true if “x” is an element in the array. If x is not in the array, return false. **This function assumes array is sorted in ascending order. You do NOT have to check if it is sorted. You can also assume the array isn’t empty.**

```
public static boolean binarySearch(int[] array, int value,
                                   int lo, int high){
    int mid = (lo+high)/2
    if (lo >= high){
      return array[lo] == value;
    }
    else if (array[mid] == value){

      return true;
    }
    else if (array[mid] < value){

      return binarySearch(array, value, mid+1, high);
    }
    else{
      return binarySearch(array, value, lo, mid-1);
    }

  }
```

1.2) (5 points) Writing Recursion

Given the following implementation of linked list:

```
public class LinkedList{
    public IntNode head;

    public LinkedList(){
        head = null;
    }

    public boolean isEmpty(){
        return (head == null);
    }

    public void insert(int x){
        //Code redacted
        //You can assume insert works
    }
}
-----------------------------------------
public class IntNode{
    public int value;
    public IntNode next;

    public IntNode(int value){
        this.value = value;
        this.next = null;
    }
}
```

Write a **recursive** method `LinkedList.length(IntNode node)` that returns the length of the list when `LinkedList.length(LinkedList.head)` is called.

If you are incapable of writing a recursive solution, you may write an iterative (looping) solution. However, you will receive an **automatic 2 point deduction** for iterative solutions.

```
public int length(IntNode node){
   if(node == null){
      return 0;
   }
   else{
      return length(node.next) + 1;
   }
}
```

(this page left intentionally blank)

2) RECURSION TRACING (10 points total)

Unfortunately, my computer was hacked by a bunch of Monty Python fans while I was working on this exam. After a vicious verbal taunting, they changed all the names of a function I had written. Below is the end result:

```
public class SpamSpamSpam{
  public static void main(String[] args){
    int ni = 0;
    char[] spamSpamSpam = args[0].toCharArray();
    System.out.println(MinistryOfSillyWalks(spamSpamSpam, ni));
  }

  //and now for something completely different

  public static boolean MinistryOfSillyWalks(char[] ni, int spam){
    if (spam>ni.length/2){
      return true; //always look on the brightside of life
    }
    int notDeadYet = ni.length - (1 + spam);
    if(ni[spam] == ni[notDeadYet]){
      return MinistryOfSillyWalks(ni, spam + 1);
      //your father smelt of elderberry
    }
    else{
      return false; //I came in here for an argument
    }
  }
}
```

To help me out, tell me what prints as a result of the following calls (use the following blank page for notes). Note that if String str = “Hello”, str.toCharArray() returns an array containing the character of str in order, i.e. [‘H’, ‘e’, ‘l’, ‘l’, ‘o’]

(2 points each)

a) java SpamSpamSpam spam false

b) java SpamSpamSpam poptop false

c) java SpamSpamSpam racecar true

d) java SpamSpamSpam Noon false

e) **(2 points)** What was the purpose of this program(roughly 10 words or less)?

Determines if the input text is a palindrome.

(This page left intentionally blank)

3.) USING OBJECTS (8 points total)

3.1) Consider the following object code to answer questions on the next page. Do not worry about the heavy mathematics in the code, you do not need to do math.

```
public class Triangle{
  public double sideA, sideB, sideC;
  public double angleA, angleB, angleC; //in degrees
  private double pi = 3.14159265;
  //sideA is OPPOSITE angleA

  public Triangle(double sA, double sB, double sC){
    sideA = sA; sideB = sB; sideC = sC;
    angleA = cosineLaw(sideA, sideB, sideC);
    angleB = cosineLaw(sideB, SideA, sideC);
    angleC = cosineLaw(sideC, sideA, sideB);
  }

  public boolean isEquilateral(){
    //TODO: incomplete method
  }

  private double radiansToDegrees(double radians){
    return radians * 180 / pi;
  }

  public double cosineLaw(double opp, double adj1, double adj2){
    double cosVal = (adj1*adj1 + adj2 * adj2 - (opp*opp)) / (2*adj1*adj2);
    return radiansToDegrees(Math.acos(cosVal));
  }
}
```

a) **(2 points)** Complete the method `isEquilateral()`. This method should return true if and only if all the sides of the triangle are equal. This should be a very small function.

```
public boolean isEquilateral(){
  return sideA == sideB && sideB == sideC;
}
```

b) **(4 points)** The following 4 lines of code are inside of **another class (not Triangle)** that uses `Triangle`. Each has a compilation error (no more than 1 on each line). Do not worry about mathematical errors, only compile time errors. Identify the errors, you do not need to correct them.

```
Triangle tri = Triangle(3,4,5);
Forgot the keyword "new" before the constructor call;


double pi = tri.pi;
pi is private, and cannot be accessed outside of Triangle;


sideA = sideB;
Cannot access sideA and sideB without referencing an instance.
I.e., tri.sideA, tri.sideB

boolean b = tri.isEquilateral;
Forgot to include the parentheses at the end of isEquilateral
```

c) **(2 points)** Some of the methods and fields should be static. List which methods and fields should be changed to static.

pi, radiansToDegress, and cosineLaw should be static.

4.) SORTING (10 points)

a) **(1 point)** What sorting algorithm is illustrated by the following (bubble, selection, insertion, merge):

Initial: 19, 61, 7, 78, 3
-----Sorting starts-----
19, 61, 7, 78, 3
7, 19, 61, 78, 3
7, 19, 61, 78, 3
3, 7, 19, 61, 78

Insertion sort

b) **(3 points)** Illustrate the steps of a merge sort with the following array to arrive at an array in ascending order. Simply show how the array changes throughout the merge sort.

{17, 12, 14, 9, 6, 1, 15, 3}

Acceptable answer

Split into arrays of size 1 and merge

[17, 12, 14, 9] [6, 1, 15, 3]
[17, 12] [14, 9] [6, 1] [15, 3]
[17] [12] [14] [9] [6] [1] [15] [3]

--begin merging
[12, 17] [9, 14] [1,6] [3, 15]
[9,12,14,17] [1,3,6,15]
[1,3,6,9,12,14,15,17]

Technically a more correct answer is:

[17,12,14,9,6,1,15,3]
[12,17,14,9,6,1,15,3]
[12,17,9,14,6,1,15,3]
[9,12,14,17,6,1,15,3]
[9,12,14,17,6,1,15,3]
[9,12,14,17,1,6,15,3]
[9,12,14,17,1,6,3,15]
[9,12,14,17,1,3,6,15]
[1,3,6,9,12,14,15,17]

c) **(1 point)** Consder a method that sorts an array in DESCENDING order. This method should (in order)

i) Find the largest value between the current point in the array and the end of the array
ii) Swap that value to the current point
iii) Increment the current point and goto i)

Which of the sorting algorithms (bubble, insertion, selection, and merge) are most like this sorting algorithm?

Selection sort

d) **(5 points)** Write the code for this sorting algorithm in Java.

```
public static void selectionSort(int[] array){
        for(int i=0; i<array.length-1; i++){
            int max = array[i];
            int maxIndex = i;
            for(int j=i+1; j<array.length; j++){
                if(array[j]>max){
                    max = array[j];
                    maxIndex = j;
                }
            }
            int temp = array[i];
            array[i] = array[maxIndex];
            array[maxIndex] = temp;
        }
}
```

5) INTERFACES (5 points)

Answer the follow questions about the interface square.

```
public interface Square{
  public double getSideLength();
  //return the length of the square's side.

  public double area();
  //return the AREA of the square

  public void draw();
  //Draw the square centered at .5 .5 in PennDraw
}
```

a) **(1 points)** What would be the FIRST line of code in a class that implements this interface (at the top of the page). Note that you can name your implementing class whatever you want.

public class MySquare implements Square{

b) **(4 points)** Explain (in roughly 30 or less words) how writing an implementation of this interface is different than writing a subclass for a parent class.

An interface is a list of methods an implementing class must contain. And interface is not functional by itself.

A parent class provides a list of methods and fields already implements. A subclass can add to or override these parent methods. A subclass is not required to implement methods. A subclass can extend the functionality of a parent class to a more specialized circumstance.

(This page left intentionally blank)

6) INHERITANCE (4 points)

Consider the class MyRectangle below.

```
public class MyRectangle{
  public double height, width;
  public double xCenter, yCenter;

  public MyRectangle(double h, double w,
                     double x, double y){
    height = h;
    width = w;
    xCenter = x;
    yCenter = y;
  }

  public void draw(){
    PennDraw.filledRectangle(xCenter, yCenter,
                             height/2.0, width/2.0);
  }
}
```

Consider a subclass of MyRectangle called MySquare.

```
public class MySquare extends MyRectangle{
  public MySquare(double w, double x, double y){
    //TODO: Write constructor
  }
}
```

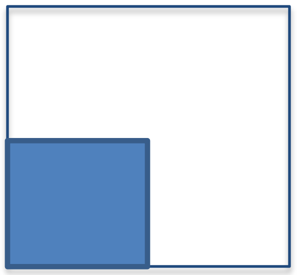
a) **(2 points)** Write the constructor for MySquare where w is the length of the squares side, x is the x value of the center of the square, and y is the y value for the center of the square. Hint: if this takes more than one line of code, you are likely doing it wrong.

super(w,w,x,y);

b) **(2 points)** What would be the result of the following lines of code given the above classes (you may either describe the result or draw a picture if the output does produce a picture):

```
MySquare ms = new MySquare(.5, .25, .25);
ms.draw();
```

```
This would draw a square in the
bottom left of a PennDraw pane.
```



7) LINKED DATA STRUCTURES (13 points)

7.1) Based on what we discussed in class, illustrate these data structures (indicating where the "head" node is for both, and tail node for queue). There is no fixed way to draw these lists, but order should be clear, as well as where the head node is.

a) **(3 points)** A stack "s" after the instructions

```
s.push(3);
s.push(7);
s.push(5);
s.pop();
s.push(6);
s.push(1);
s.pop();
```

```
top of stack ->   6,7,3
```

b) **(3 points)** A queue "q" after the instructions

```
q.queue(3);
q.queue(7);
q.queue(5);
q.dequeue();
q.queue(6);
q.queue(1);
q.dequeue();
```

```
front of queue -> 5, 6, 1 <-back of queue
```

7.2) Using the following code, answer questions on the next page:

```
public class DoublyLinkedList{
    public DLLNode head;
    public DLLNode tail;

    public DoublyLinkedList(){
        head = null;
        tail = null;
        size = 0;
    }

    public boolean isEmpty(){
        return (head == null);
    }

    public void insertAtTail(int x){
        DLLNode newNode = new DLLNode(x);
        if (isEmpty()){
            head = newNode;
            tail = newNode;
        }
        //TO DO: Insert at tail if list isn't empty
    }

    public void decapitate(){
        //replace the head node with the node after head
        //TODO
    }
}
```

```
----------------------------------------------------------------
```

```
public class DLLNode{
    public int value;
    public DLLNode next;
    public DLLNode prev;

    public DLLNode(int x){
        value = x;
        next = null;
        prev = null;
    }
}
```

a) **(1 point)** What data structure have we talked about in class that this DoublyLinkedList is most like (assuming we don't add any new methods):

queue

b) **(3 points)** Finish the function `insertAtTail(int x)` such that it adds the value x to a new node at the tail of the list. I've already written the easy part for you, which is when you insert into an empty doubly linked list. You only need to write the portion where you insert into this list at the tail.

```
public void insertAtTail(int x){
   DLLNode newNode = new DLLNode(x);
   if (isEmpty()){
        head = newNode;
        tail = newNode;
   }
    //Your Code goes here
   else{
        tail.next = newNode
        newNode.prev = tail;
        tail = newNode;
     }
```

c) **(3 points)** Write the function `decapitate()`. As its name implies this function removes the head from the LinkedList. Remove the head and make the next node after head the new head.

```
public void decapitate(){
   DLLNode oldHead = head;
   oldHead.next.prev = null;
   head = oldHead.next;
   oldHead.next = null; //technically not necessary.
}
```

8) BINARY SEARCH TREES (9 points)

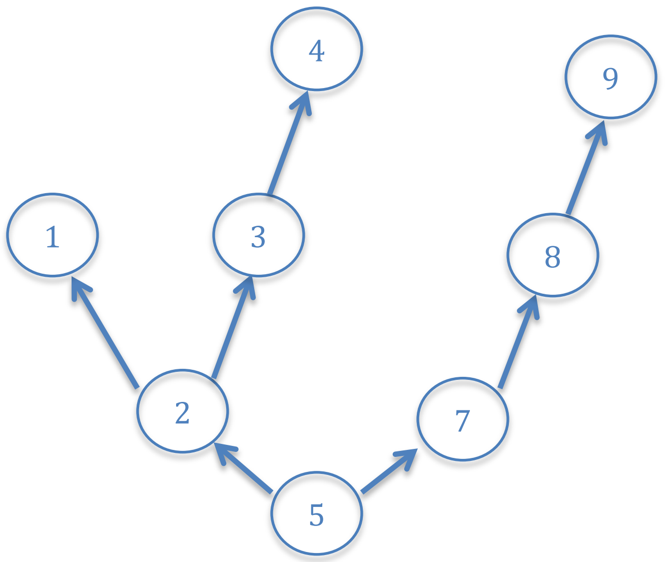
Given the binary search tree written below, answer the following questions:

```
public class BinarySearchTree{//The tree
    public BSTNode head;//"root" of the tree

    public BinarySearchTree(){
        head = null;
        size = 0;
    }

    public void insert(int x){
        //Code redacted, you can assume this
        //functions works as we discussed in class
    }

    public int getSize(){
        return size;
    }

    public boolean find(int x){
        //TODO - write find
    }
}
```

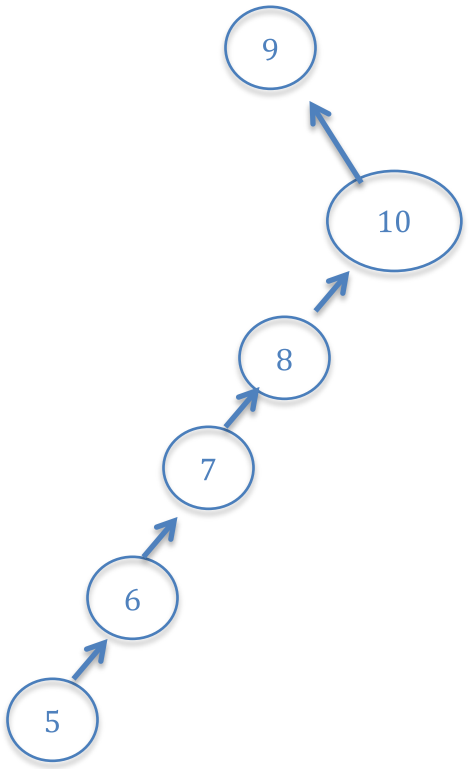
```
public class BSTNode{
    public int value;
    public BSTNode left; //less than side
    public BSTNode right; //greater than side

    public BSTNode(int value){
        this.value = value;
        left = null;
        right = null;
    }
}
```

8.1) **(6 points)** Draw what the binary search tree would look like if the following values were inserted into the tree in order from left to right:

a) 5, 7, 2, 1, 3, 4, 8, 9



b) 5, 6, 7, 8, 10, 9, 5



8.2) **(6 points)** Write the rest of the `find(int x)` function. This function takes as input an integer X and returns as output `true` if x is in the binary search tree, or `false` if x is not. If the tree is empty, this should return false (since x obviously isn't in an empty tree). You should NOT search ALL values in the Tree. Rather, use the knowledge that the tree is a functioning Binary Search Tree to write `find(int x)`.

```
public boolean find(int x){
   BSTNode search = head;
   while(true){
      if (search == null){

           return  false; //Fill in the blank
      }
      else if (search.getValue() == x){

           return true;
      }

      else if (search.getValue() > x){
           search = search.left;
      }

      else {
           search = search.right;
      }
```

(This page left intentionally blank)