CIS 110 Fall 2016 — Introduction to Computer Programming
8 Dec 2016 — Final Exam
Answer Key

1.) PennKey, Penn Mascot, etc. (2 points total)

(a) Check to make certain that your exam has all 10 pages (excluding the cover sheet).

(b) Write your name, recitation number, and PennKey (username) on the front of the exam.

(c) Sign the certification that you comply with the Penn Academic Integrity Code.

(d) Which Ivy League university has the Quaker as its mascot? Circle your answer.

- [✓ **University of Pennsylvania (Go Quakers!)**]
- Brown University
- Columbia University
- Cornell University
- Dartmouth College
- Harvard University
- Princeton University
- Yale University

2.) Brown's Pass/Fail Policy (9 points total)

In 1969, Brown University began doing away with letter grades, instead choosing to go with a pass/fail system. On Brown's computer science final, to accommodate their grade policy, students were asked to write down one thing they learned that semester. Those who wrote something true would PASS, those who wrote something false would FAIL. To avoid any appearance of favoritism from the faculty, Brown University outsourced the exam grading to Dr. Brown at Penn. He, in return, is delegating it to you.
Help Dr. Brown grade Brown University's exams by determining if the student who wrote each sentence gets a PASS (true) or a FAIL (false).

Student's Submission	**Pass/Fail**
Encapsulation is the idea that more information isn't always better.	Pass
MergeSort is called MergeSort because it merges the best parts of a selection sort and an insertion sort.	Fail
Helper functions in object oriented design should be made public.	Fail
An "instance" of an object is a version of the object used only in a local scope, i.e. instantly.	Fail
A key advantage of a linked list over an array is resizability.	Pass
A Stack is First In, Last Out.	Pass
An interface lists and implements a set of functions for a class.	Fail
Linked lists have a faster access time than arrays, which is why you should usually use Linked Lists.	Fail
If you do not write a constructor for an object, all primitive data types in your object will be initialized to null.	Fail

3.) Hunting Clowns at Yale (8 points total)
Little known fact: Dr. McBurney hates clowns almost as much as Joe Biden loves ice cream. That's why he took a year-long trip to Yale — the top clown college in the world — to hunt the wretched beasts. (Crime-ridden New Haven, CT has been trying to eliminate the clown mafia for decades, with little success.) During his year at Yale, Dr. McBurney put together a clown tracking program, but some of the code was erased when the TSA mishandled his laptop before the flight home.
A clown map is a boolean array consisting of only `trues` and `falses`. A clown den in the map is defined as a series of 2 or more adjacent `trues`. In the code below, fill in the blanks to complete the `getClownDenCount()` method that returns the number of Clown dens present in the map field.

Examples:

```
boolean[] map = {true, false, false, false, false};
ClownSlayer will = new ClownSlayer(map);
will.getClownDenCount() returns 0
```

```
boolean[] map = {true, true, true, false, true, true};
ClownSlayer will = new ClownSlayer(map);
will.getClownDenCount() returns 2
```

```
public class ClownSlayer {
    boolean[] map;

    public int getClownDenCount() {
        // error check
        if (map == null) {
            return 0;
        }

        int count = 0;
        boolean foundDen = false;
        for (int i = 0; i < map.length - 1; i++) {
            if (map[i] && map[i + 1] && !foundDen) {
                foundDen = true;
                count++;
            } else if (!map[i]) {
                foundDen = false;
            }
        }
        return count;
    }

    // ... the rest of ClownSlayer is here
}
```

4.) Grade Inflation (22 points total)
Harvard is teaching a computer science class this semester, but even very lenient grading isn't yielding enough As. The unversity has therefore decided to give the top student a 100%, and give all other students the score earned by the student scoring immediately higher. Harvard asked the students in the class to write a function that would compute the revised scores, but the students did a lousy job. So now Harvard wants you to figure out what their students wrote and fix it.

For example, if the student grades were

`int[] studentGrades = { 40, 80, 60, 90, 30 }`

Then the function is supposed to modify the grades to be `{ 60, 90, 80, 100, 40 }`.

Sometimes multiple students have the same grade, for instance:

`int[] studentGrades = { 60, 40, 40, 90 }`

Then both the 40s should get bumped up to the next higher grade: `{ 90, 60, 60, 100 }`.

4.1) (2 points) To start off, the Harvard students wrote a sorting function. It does work, but the university wants some more information about it:

```
public static void sort(int[] array) {
    for (int i = 0; i < array.length - 1; i++) {
        int a = array[i];
        int b = i;
        for (int j = i + 1; j < array.length; j++) {
            if (array[j] < a) {
                a = array[j];
                b = j;
            }
        }
        int temp = array[i];
        array[i] = array[b];
        array[b] = temp;
        printArray(array);
    }
}
```

(a) What is the name of this sorting algorithm? Selection Sort

(b) Does this function sort in ascending (smallest to largest) or descending (largest to smallest) order? Ascending

QUESTION CONTINUES ON THE NEXT PAGE

Grade Inflation (Continued)

Using the sorting function from the previous page, the Harvard students created the following buggy function to raise all their grades (line numers have been added so you can refer to lines of code easily):

```
 1. public void inflateGrades(int[] studentGrades) {
 2.     int[] copyOfGrades = int[studentGrades.length];
 3.
 4.     // copy the contents of studentGrades 1 at a time
 5.     copyOfGrades = studentGrades;
 6.
 7.     copyOfGrades = sort(copyOfGrades);
 8.
 9.     // for each grade in student grades...
10.     for (int i = 0; i <= studentGrades.length; i++) {
11.         // if student has maximum grade, give 100
12.         if (studentGrades[i] == copyOfGrades[0]) {
13.             studentGrades[i] = 100;
14.             break;
15.         }
16.
17.         // otherwise find the next grade after toBeFound
18.         int toBeFound = studentGrades[j];
19.         for (int j = 0; j <= studentGrades.length; j++) {
20.             if (toBeFound == copyOfGrades[j]) {
21.                 while(toBeFound == copyOfGrades[j])
22.                     j--;
23.                 }
24.                 studentGrades[j] = copyOfGrades[j];
25.             }
26.         }
27.     }
28. }
```

On the following page, identify each buggy line in this program, and explain the bug in **20 words or less**. If you feel a single bug is rooted in multiple lines of code, list all lines in the same box. You may either explain the bug without fixing it, or provide a snippet of code that would correct it. You do not need to write complete sentences or complete corrected lines of code. **There may be fewer bugs than rows in the table!**

QUESTION CONTINUES ON THE NEXT PAGE

Grade Inflation (Continued)

4.2) (20 points) Identify each buggy line in the program on the previous page, and explain the bug in **20 words or less**. If you feel a single bug is rooted in multiple lines of code, list all lines in the same box. You may either explain the bug without fixing it, or provide a snippet of code that would correct it. You do not need to write complete sentences or complete corrected lines of code. **There may be fewer bugs than rows in the table!**

Line Number	Brief description of bug (<= 20 words)
2	Missing `new` keyword
5	Makes reference, doesn't copy values, so `copyOfGrades` sorts boths arrays
7	`sort` is `void`, so cannot assign result to a variable
10	Need to use `< studentGrades.length` or `<= studentGrades.length - 1`
12	Maximum value is in `copyOfGrades[copyOfGrades.length - 1]`
14	Need `continue` instead of `break`
18	Need `studentGrades[i]`, not `j`
19	Need to use `< studentGrades.length` or `<= studentGrades.length - 1`
22	Need `j++`, not `j--`
24	Need `studentGrades[i]`, not `j`

5.) Animal House (12 points total)
Members of Animal House's Delta fraternity (modeled on life at Dartmouth College) constantly face expulsion over their low GPAs. Recently, some of them have written a (poor) program to manipulate their GPAs, and the dean wants to know exactly what it does. To that end, he has asked you to figure out the values of certain variables at each of four points in the program:

```
public class DartmouthStudent {
    private double gpa;
    private String nickname;

    public DartmouthStudent(double g, String n) {
        gpa = g;
        nickname = n;
    }

    public double getGPA()            { return gpa;      }
    public void setGPA(double gpa)    { gpa = this.gpa;  }
    public String getNickname()       { return nickname; }
    public void setNickname(String n) { nickname = n;    }

    public static void fixGPA(double g, DartmouthStudent delta) {
        g = 1.0 / 0.0;
        double gpa = delta.getGPA();
        delta.setNickname("qwerty");
        /* POINT B */
        delta.setGPA(1.6);
        delta = new DartmouthStudent(0.678, "Bluto");
        /* POINT C */
    }

    public static void main(String[] args) {
        double g = 0.05;
        DartmouthStudent delta = new DartmouthStudent(0.08, "Otter");
        /* POINT A */
        fixGPA(g, delta);
        System.out.print(delta.getNickname());
        /* POINT D */
    }
}
```

	`g`	`delta.gpa`	`delta.nickname`
Point A	0.05	0.08	Otter
Point B	`Infinity`	0.08	qwerty
Point C	`Infinity`	0.678	Bluto
Point D	0.05	0.08	qwerty

6.) The Hottest Holiday TOY (15 points total)

Because of their proximity to 5th Avenue, Columbia students have an edge over all other Ivy Leaguers in finding out what the hot toy for the holidays will be and in getting hold of it. But this year, they decided it would be ironic to sit around in a dim cafe and play with TOY instead. In one recent afternoon, Columbia students produced the following gem, although they didn't add any comments of their own. As a result, we don't know what output it produces, or even how many values it outputs.

```
01: 0003   (0000 0000 0000 0011,      3)
10: 7301   R[3] <- 0001
11: 5133   R[1] <- R[3] << R[3]
12: 8201   R[2] <- mem[01]
13: 8415   R[4] <- mem[15]
14: D116   if (R[1] > 0) goto 16
15: 0000   halt
16: 93FF   write R[3]
17: 94FF   write R[4]
18: D415   if (R[4] > 0) goto 15
19: 94FF   write R[4]
1A: 2A12   R[A] <- R[1] - R[2]
1B: DA01   if (R[A] > 0) goto 01
1C: 91FF   write R[1]
1D: 5332   R[3] <- R[3] << R[2]
1E: 143A   R[4] <- R[3] + R[A]
1F: C415   if (R[4] == 0) goto 15
20: 94FF   write R[4]
21: D301   if (R[3] > 0) goto 01
22: 93FF   write R[3]
23: 0000   halt
```

6.1) (15 points) What values does this program write to standard output? Write your answer as a list of numbers in ordinary, base 10 notation.

1, 0, 0, 2, 7 [Editor's Note: Apparently the program just print's out Columbia's ZIP code.]

7.) Surfing the Dinky (24 points total)

A particulary reprehensible (and thoroughly illegal) Princeton tradition is "surfing the Dinky," which involves riding on top of the train connecting Princeton with the main station at Princeton Junction. There are a number of points along this short journey where you are liable to get electrocuted by a low-hanging catenary wire or bludgeoned by a low-hanging tree branch or beam. In a seriously misguided effort to "improve" the safety of this activity, you propose compiling a list of when to duck to avoid obstacles. More precisely, you suggest recording how many seconds should pass before you have to duck again. Since you would need to refine this list over a few test runs, and update it as overhead lines and obstructions are modified, you decide to implement this with a linked list using the following node class:

```
public class DuckNode {
    public int interval;  // duck after interval seconds
    public DuckNode next; // next node in the list
}
```

We'll skip over the part where you construct this list, since surfing the dinky is a really, really bad idea, and instead try and salvage the list that has been so painfully created for legitimate purposes. For example, you could figure out the total travel time from this list, since you would need to duck as the train arrives at the covered station at either end. You can also figure out the average travel time between low-hanging obstacles (total travel time divided by number of obstacles), which might be useful if NJ Transit ever decides to introduce a double-decker dinky.

7.1) (7 points) Write a public static function `iterativeTravelTime` that is part of the `DuckNode` class that takes a single `DuckNode` argument and returns the total travel time as an integer. Return 0 if the argument is null. The function must be iterative, and may not call any functions or methods. Do not write "`class DuckNode`" around the function, just write that function itself. You do not have to write any comments, but you are welcome to do so.

```
public static int iterativeTravelTime(DuckNode head) {
    if (head == null) return 0;
    int travelTime = 0;
    for (DuckNode n = head; n != null; n = n.next) {
        travelTime += n.interval;
    }
    return travelTime;
}
```

QUESTION CONTINUES ON THE NEXT PAGE

Surfing the Dinky (Cont'd)

7.2) (7 points) Write a public non-static method `recursiveTravelTime` that is part of the `DuckNode` class that takes no arguments and returns the total travel time as an integer. This version must be recursive, may not call any *other* functions or methods, and many not contain any loops.

```
public int recursiveTravelTime() {
    if (next == null) return interval;
    return interval + next.recursiveTravelTime();
}
```

7.3) (10 points) Write a public static function `avgTimeBetweenObstacles` that is part of the `DuckNode` class that takes a single `DuckNode` argument and returns the *average* time between obstacles as a double. Return 0 if the argument is null. This function must be self-contained and may not call either of your previous two methods. It can be recursive or iterative or a combination of both. You may write **at most one** private helper method (you don't have to), which can be static or non-static as you prefer.

```
public static double avgTimeBetweenObstacles(DuckNode head) {
    if (head == null) return 0;

    int count = 0;
    int travelTime = 0;
    for (DuckNode n = head; n != null; n = n.next) {
        travelTime += n.interval;
        count++;
    }

    return travelTime / (double) count;
}
```

Note: This is extremely difficult or impossible to implement recursively within the constraints of the question because you need to compute both the total travel time and the number of nodes in order to compute the average. That would either require two recursive helper methods, one for each value, or some sort of inelegant gimmick effectively return two values at a time.

TOY Reference Card

```
INSTRUCTION FORMATS

             | . . . . | . . . . | . . . . | . . . .|
   Format 1: | opcode  |    d    |    s    |    t   |  (0-6, A-B)
   Format 2: | opcode  |    d    |       addr       |  (7-9, C-F)


ARITHMETIC and LOGICAL operations
     1: add              R[d] <- R[s] +  R[t]
     2: subtract         R[d] <- R[s] -  R[t]
     3: and              R[d] <- R[s] &  R[t]
     4: xor              R[d] <- R[s] ^  R[t]
     5: shift left       R[d] <- R[s] << R[t]
     6: shift right      R[d] <- R[s] >> R[t]

TRANSFER between registers and memory
     7: load address     R[d] <- addr
     8: load             R[d] <- mem[addr]
     9: store            mem[addr] <- R[d]
     A: load indirect    R[d] <- mem[R[t]]
     B: store indirect   mem[R[t]] <- R[d]

CONTROL
     0: halt             halt
     C: branch zero      if (R[d] == 0) pc <- addr
     D: branch positive  if (R[d] >  0) pc <- addr
     E: jump register    pc <- R[d]
     F: jump and link    R[d] <- pc; pc <- addr


Register 0 always reads 0.
Loads from mem[FF] come from stdin.
Stores to mem[FF] go to stdout.
```