CIS 110 — Introduction to Computer Programming

27 June 2013 — Final Exam

Answer Key

Miscellaneous

0. (1 point)

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

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

True/False

1. (5 points)

Say whether each of the statements below is **True**, **False**, or **Sometimes** true and sometimes false.

(a) `Array`s have a method named `length` that returns their size.

Answer: False

(b) The default value for `boolean` variables is `false` if you don't explicitly initialize them.

Answer: Sometimes

(c) Each class can have only one constructor.

Answer: False

(d) It would make more sense to use a linked list than an array for the `RingBuffer` object in the Guitar Hero assignment.

Answer: False

(e) Linked list elements are always stored sequentially in memory.

Answer: False

Short Answer

2. (12 points)
Answer each of the three questions below and on the next page:

(a) List all combinations of method signatures below that could ***not*** simultaneously be part of the same class. Draw a box around your answer so we can distinguish it from your scratch work.

i. `public int add(Node foo, Node bar, Node foobar)`
ii. `public int add(Node b, int a, Node c)`
iii. `public int add(int c, Node b, Node a)`
iv. `public int add(Node b, Node d, Node foo)`
v. `public int add(Node bar, Node foobar)`
vi. `public double add(Node a, Node b, Node c, Node d)`
vii. `public double add(int a, Node c, Node b)`
viii. `private int add(Node head, int x, Node tail)`

i. and iv.
ii. and viii.
iii. and vii.

(b) Consider a stack of integers `s` with a `push()` method to add elements and a `pop()` method to remove and return elements. Assuming `s` starts out empty, fill in the blanks in the code sequence below so it is correct.

```
s.push(4);
s.push( 6     );
s.push(8);
s.pop(); returns  8
s.push(6);
s.push( 4     );
s.push(7);
s.pop(); returns 7
s.pop(); returns 4
s.pop(); returns  6
s.pop(); returns 6
s.pop(); returns  4
```

(c) Give one example (10 words or less) of a situation where you would use a more complex data structure than a linked list, array, single object, or variable. *Hint: This is a short answer question, don't over-think it.*

Examples include a tree (e.g. a family tree or binary search tree), a symbol table (e.g. a telephone book), and a graph (e.g. a maze).

Amaze Us (2 Pages)

3. (10 points)

(a)
```
public static int A(Vertex v) {
  int a = 0;
  for (EdgeListNode n = v.edges.next; n != null; n = n.next)
    a++;
  return a;
}
// Returns the number of edges leaving vertex v.
```

(b)
```
public double B() {
  double b = 0;
  for (int i = 0; i < rooms.length; i++)
    b += A(rooms[i]);
  return b / rooms.length;
}
// Returns the average number of edges leaving each vertex in this graph.
```

(c)
```
public boolean C(int x, int y) {
  for (EdgeListNode n = rooms[x].edges.next; n != null; n = n.next)
    if (n.target == y) return false;
  return true;
}
// Returns true if there is no edge from vertex x to vertex y.
```

(d)
```
public int[] D() {
  int[] d = new int[rooms.length];
  for (int i = 0; i < d.length; i++)
    for (EdgeListNode n = rooms[i].edges.next; n != null; n = n.next)
      d[n.target]++;
  return d;
}
// Return an array of the number of edges entering each vertex.
```

(e)
```
public void E(int x, int y) {
  EdgeListNode e = rooms[x].edges;
  while (e.next != null) e = e.next;
  e.next = new EdgeListNode();
  e.next.target = y;
}
// Adds an edge from vertex x to vertex y to the end of x's list of edges.
```

(f)
```
public void F() {
  for (int i = 0; i < rooms.length; i++)
    for (EdgeListNode n = rooms[i].edges.next; n != null; n = n.next)
      if (C(n.target, i)) E(n.target, i);
}
/* Adds an edge from y to x whenever it doesn't exist but the edge from x to
y does. — or — Adds the back edges for all existing edges, without adding
duplicates. — or — Turns a directed graph into an undirected graph. */
```

Let's Be Leakers

4. (18 points) Assume you call the `manning` function below with three integer arguments. For each of the six points indicated in the program, say whether the listed conditions are sometimes, always, or never true. Write **S**, **A**, and **N** for sometimes, always, and never.

```
public class Leakers {
  private static int x, y, z;

  public static void manning(int x, int y, int z) {
    z = y * y;
    if (y < 0) y = -y;
    y++;
    x = x / y;
    // Point A
    if (x < 0)      z = y + x;
    else if (x > 0) z = y - x;
    else            z = y;
    // Point B
    z = snowden(2 * y + Math.abs(x), y, z);
    y = snowden(z, 2 * z, y);
    // Point C
    if (x < y) {
      x = ellsberg(2 * z, z, y);
      y = x;
      // Point D
    } else {
      x++;
      y = y - x;
      // Point E
    }
    // Point F
  }

  private static int snowden(int x, int y, int z) {
    while(x > y) {
      z++;
      x--;
    }
    return z;
  }

  private static int ellsberg(int x, int y, int z) {
    if (x > y) return ellsberg(x - 1, y, z) + 1;
    else       return z;
  }
}
```

	z > y	x > 0	y >= 0
A	S	S	A
B	N	S	A
C	A	S	A
D	N	A	A
E	A	A	N
F	S	A	S

Link, Don't Blink!

5. (24 points)

(a) Write a constructor for `DLinkedList` that takes an array of integers and inserts them in order into the double linked list. If the array is null or empty, the list should remain empty.

```
public DLinkedList(int[] arr) {
  // if the array is null or empty, return the empty list
  if (arr == null || arr.length == 0) return;

  // add the first element to the list (head == tail in this case)
  head = new DNode();
  head.val = arr[0];
  tail = head;

  // add the remaining elements onto the end
  for (int i = 1; i < arr.length; i++) {
    tail.next = new DNode(); // put new node on the end
    tail.next.prev = tail;   // new node comes after current tail
    tail.next.val = arr[i];  // set new node's value
    tail = tail.next;        // update the tail
  }
}
```

(b) Write a method, `reverse`, for the `DLinkedList` class that reverses the list's direction so the head becomes the tail and vice versa. For full credit you must not use the `new` keyword.

```
public void reverse() {
  // reverse the links on every node
  // the update is n = n.prev because it is executed
  // after we swap the next and prev pointers
  for (DNode n = head; n != null; n = n.prev) {
    // swap next and prev
    DNode tmp = n.next;
    n.next = n.prev;
    n.prev = tmp;
  }

  // now swap head and tail
  DNode tmp = head;
  head = tail;
  tail = tmp;
}
```