CS/ENGRD2110: Prelim 2
SOLUTION

17th of April, 2012

NAME : _____

NETID: _____

- The exam is **closed book and closed notes**. Do not begin until instructed. You have **90 minutes**. Good luck!
- Start by writing your name and Cornell netid on top! There are **11 numbered pages**. Check now that you have all the pages.
- Web, email, etc. may not be used. Calculator with programming capabilities are not permitted. This exam is **individual work**.
- We have **scrap paper** available, so you if you are the kind of programmer who does a lot of crossing out and rewriting, you might want to write code on scrap paper first and then copy it to the exam, just so that we can make sense of what you handed in!
- Write your answers in the space provided. Ambiguous answers will be considered incorrect. You should be able to **fit your answers easily into the space we provided**. Answers that are not concise might not receive full points. It you do need more space, use the back page of the exam.
- In some places, we have abbreviated or condensed code to reduce the number of pages that must be printed for the exam. In others, code has been obfuscated to make the problem more difficult. This does not mean that its good style.

POINTS:

Dictionaries and Hashtables	_____ / 12
Stacks/Queues	_____ / 15
Priority Queues and Heaps	_____ / 16
Graphs	_____ / 25
Graph Search	_____ / 12
Graphical User Interfaces	_____ / 11
	==========
Total	_____ / 91

1 Dictionaries and Hashtables

1. If you were given the job to write a hash function where they key is a linked list of integers. Denoting the list as (a,b,c,d,...), which one of the following hash functions is the BEST:

   2 pts.

   (a) return the value of Math.Random (i.e. a random number) rounded to the closest integer

   (b) return a

   (c) return min(a,b,c,d,...)

   (d) if the elements of the linked list are (a,b,c,d,...), return a*b*c*d*...

   SOLUTION:
   (a) is not deterministic. (b) all lists that start with the same number have the same hash value, so there are many collisions and only the first elements of the array will be used independent of the overall list size. (c) analogous problem to (b). (d) is the best solution, since it best spreads out the hash keys.
   END SOLUTION

2. Mark all properties that are TRUE for a hashtable with n elements?

   4 pts.

   (a) an ideal hash table using array doubling has worst-case time complexity of $O(1)$ for every insert operation

   (b) an ideal hash table using array doubling has average-case time complexity of $O(1)$ for lookups

   (c) can be used to sort an array of n real numbers with average-case time complexity $O(n)$

   (d) it is possible to have different keys being hashed to the same position in the array

   SOLUTION:
   (b) and (d) are true
   END SOLUTION

3. We have a hash table of size 7 to store integer keys, with **linear probing** and a hash function h(x) = x mod 7 (x mod 7 return the remainder of the integer division with 7). Show the content of the hashtable after inserting the keys 0,11,3,7,1,9 in the given order.

   6 pts.

   SOLUTION:

Index	0	1	2	3	4	5	6
Key	0	7	1	3	11	9	

   END SOLUTION

2 Stacks and Queues

1. Consider an implementation of a queue using an array with wraparound and table doubling to handle overflow. The array size is initially 3, and the queue is empty. Draw the contents of the array after each of the following operations. For the poll() operations, also give the return value.

   7 pts.

   SOLUTION:

   - initially:

Index	0	1	2
Value			

   - add(5):

Index	0	1	2
Value	5		

   - add(3):

Index	0	1	2
Value	5	3	

   - poll():

Index	0	1	2
Value	5	3	

     , return = 5

   - add(4):

Index	0	1	2
Value	5	3	4

   - poll():

Index	0	1	2
Value	5	3	4

     , return = 3

   - add(6):

Index	0	1	2
Value	6	3	4

   - add(7):

Index	0	1	2
Value	6	7	4

   END SOLUTION

2. In class you saw an algorithm that implements a queue using two stacks A and B. Give the best big-O complexity for the following questions regarding the two-stack queue, assuming that you add/poll n elements in total:

   8 pts.

   (a) What is the worst-case time complexity of each "add(x)".

   (b) What is the worst-case time complexity of each "poll()".

   (c) What is the average time complexity of an "add(x)" amortized over n operations.

   (d) What is the average time complexity of a "poll()" amortized over n operations.

   SOLUTION:
   O(1), O(n), O(1), O(1)
   END SOLUTION

3 Priority Queues and Heaps

1. Mark the following statements as either TRUE or FALSE:

   4 pts.

   (a) HeapSort has better worst-case time complexity than MergeSort (in term of big-O)

   (b) HeapSort can sort an array in place (i.e. does not need extra memory)

   (c) A heap can be used to implement a queue

   (d) A heap can be used to implement a stack

   SOLUTION:
   FALSE, TRUE, TRUE, TRUE
   END SOLUTION

2. The worst-case time complexity of HeapSort is:

   2 pts.

   (a) $O(log(n))$

   (b) $O(n)$

   (c) $O(nlog(n))$

   (d) $O(n^2)$

   SOLUTION:
   (c)
   END SOLUTION

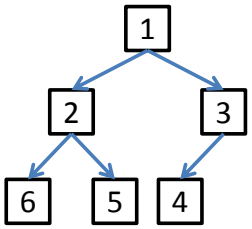
3. Construct a balanced binary min-heap (i.e. a heap that always returns the minimum element) using the following elements, pushing them onto the heap in the given order:

   6, 4, 2, 1, 5, 3

   Draw the heap after EACH completed insertion of an element.

   6 pts.

   SOLUTION:
   This is the correct final heap:

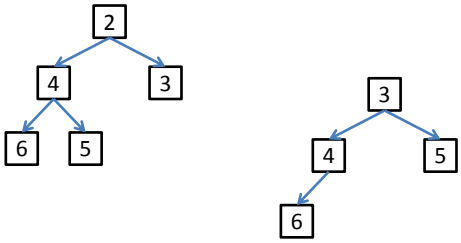


   END SOLUTION

4. Now pop (i.e. extract) the two smallest elements off the heap. Draw the heap after each such extraction.

4 pts.

SOLUTION:
These are the correct heaps:



END SOLUTION

4 Graphs

1. Let G be a directed graph with a finite number of nodes. Are the following statements TRUE or FALSE?

   4 pts.

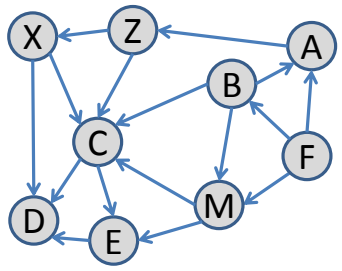
   (a) If G is directed acyclic, then there is a vertex with no incoming edges.

   (b) If G is directed acyclic, then there is a vertex with no outgoing edges.

   SOLUTION:
   TRUE, TRUE
   END SOLUTION

2. How many cycles does the following directed graph contain?

   3 pts.



   SOLUTION:
   zero
   END SOLUTION

3. Perform topological sort on the graph from above and write down the sorted list of nodes. If there is more than one valid topological sort order, write down all of them.

   4 pts.

   SOLUTION:
   F,B,M,A,Z,X,C,E,D or
   F,B,A,M,Z,X,C,E,D or
   F,B,A,Z,M,X,C,E,D or
   F,B,A,Z,X,M,C,E,D
   END SOLUTION

4. In the graph from the previous question, which nodes are reachable from node Z?

   4 pts.

   SOLUTION:
   (Z),X,C,D,E
   END SOLUTION

5. For the same graph, write down all paths from node A to node D?

   4 pts.

   SOLUTION:
   A,Z,X,D
   A,Z,X,C,D

A,Z,X,C,E,D
A,Z,C,D
A,Z,C,E,D
END SOLUTION

6. Consider the graph G with vertices V = { 1, 2, 3, 4 } and edges $E = \{(1,2),(2,3),(3,4),(4,1),(2,1),(2,4)\}$. For every vertex u, find its indegree in(u) and its outdegree out(u). What is the value of the following sum for this graph?

$$\sum_{u \in V} [in(u) + out(u)] =$$

3 pts.

SOLUTION:
12
END SOLUTION

7. How does the value of the following sum depend on n and m for any graph with n vertices and m edges? Note that the sum is different from the sum in the last question!

$$\sum_{u \in V} [in(u) - out(u)] =$$

3 pts.

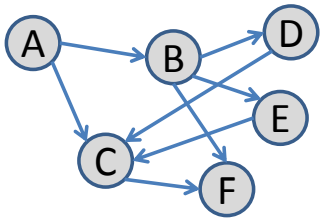
SOLUTION:
0
END SOLUTION

5 Graph Search

1. A bipartite graph is a graph that is 2-colorable. This means that all nodes can be painted either GREEN or BLUE such that edges go only between nodes of different color.

   Is the following graph bipartite? If yes, show the coloring of the nodes that proves it. If no, show which node needs a third color.

   5 pts.



   SOLUTION:
   yes. B,C=GREEN, A,D,E,F=BLUE
   END SOLUTION

2. Give a pseudo-code implementation of a method `bipartite(G)` that returns true if the undirected graph G is bipartite, and returns false otherwise. Indicate which datastructures you are using. You can assume that standard datastructures are available, and that G is connected. For full credit, describe an algorithm that has worst-case runtime complexity $O(m)$, where m is the number of edges in G.

   7 pts.

   SOLUTION:
   One can use the following version of BFS (or DFS if one replaces the queue with a stack):

   - initialize queue Q as empty
   - initialize hashtable C for storing the color of each node as empty
   - put random node n from G into Q
   - color n as GREEN in C
   - WHILE(Q not empty)
     - n = Q.extract()
     - FORALL neighbors n' of n in G
       - IF n' has no color yet in C THEN color n' as the opposite color of n and add n' to Q
       - IF n' has same color as n in C THEN return FALSE
   - return TRUE

   END SOLUTION

6 Graphical User Interfaces

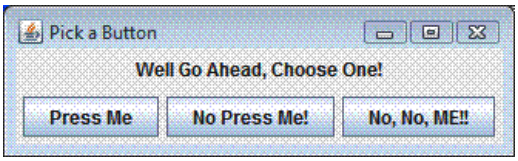
For all questions in this section, refer to the following program:

```
import javax.swing.*;import java.awt.*;import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class GUIQuestion extends JFrame {
        JButton b1, b2, b3;
        JPanel text;
        JLabel choose, chosen;

        public GUIQuestion(){
                super("Pick a Button");
                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                setSize(500, 500);

                text= new JPanel(new FlowLayout (FlowLayout.CENTER));
                choose= new JLabel("Well Go Ahead, Choose One!");
                text.add(choose);

                JPanel buttons= new JPanel(new FlowLayout (FlowLayout.CENTER));
                b1= new JButton("Press Me");
                b2= new JButton("No Press Me!");
                b3= new JButton("No, No, ME!!");
                buttons.add(b1);
                buttons.add(b2);
                buttons.add(b3);

                add(text, BorderLayout.NORTH);
                add(buttons, BorderLayout.SOUTH);
                setVisible(true);
                pack();

                b1.addActionListener(new ActionListener(){
                        public void actionPerformed(ActionEvent e) {
                                text.remove(choose);
                                text.add(new JLabel("Thank you for choosing 1!"));
                                pack();
                        }
                });
                b2.addActionListener(new ActionListener(){
                        public void actionPerformed(ActionEvent e) {
                                text.add(new JLabel("2 was chosen. "));
                                pack();
                        }
                });
                b3.addActionListener(new ActionListener(){
                        public void actionPerformed(ActionEvent e) {
                                text.add(new JLabel("You picked 3. "));
                                b1.setText("Don't forget about me!");
                                b2.setText("I guess I just got my turn.");
                                b3.setText("Finally, thank you!");
                                pack();
                        }
                });
        }

        public static void main(String[] args){
                new GUIQuestion();
        }
}
```

1. Draw a sketch of what the GUI looks like before any interaction with the user. The sketch should include all features of the GUI and should put them roughly in the correct position.

   6 pts.

   SOLUTION:

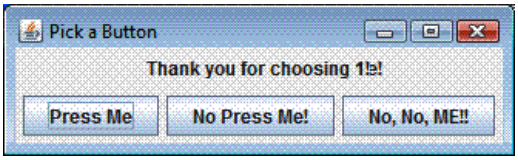


   END SOLUTION

2. Describe or sketch how the window changes after the user has clicked the button "b1"?

   2 pts.

   SOLUTION:

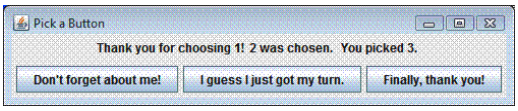


   The text "Well Go Ahead, Choose One!" gets removed and instead "Thank you for choosing 1!" gets inserted.
   END SOLUTION

3. Describe or sketch how the window changes after the user has now also clicked the button "b2" followed by "b3"?

   3 pts.

   SOLUTION:



   The text "2 was chosen. You picked 3" gets inserted after the text "Thank you for choosing 1!". The buttons are relabeled.

   END SOLUTION