Prelim 2[Solutions]

1. Short Answer [18 pts]

   (a) [3 pts] In a model/view/controller, Swing is likely to be part of (there may be more than one):

   i. the model
   ii. the view
   iii. the controller

   Answer:

   The view and the controller.

   (b) [4 pts] Which of the following must be true (there may be more than one)?

   i. if x.hashCode() == y.hashCode() then x.equals(y)
   ii. if x.hashCode() == y.hashCode() then x == y
   iii. if x.equals(y) then x.hashCode() == y.hashCode()
   iv. if x == y then x.hashCode() == y.hashCode()

   Answer:
   The first and second need not be true. The third must be true because otherwise hash table lookups will not find entries that are equal. The fourth must be true because hashCode should return the same result if the object hasn't changed.

   (c) [4 pts] A traversal of an expression tree produces the string "$+ + 3\ 5 * 4\ 2$".

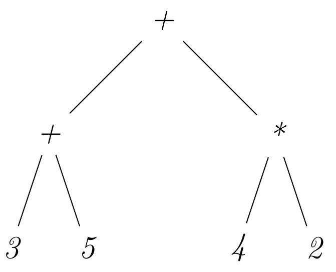
   i. What kind of traversal is it?

   Answer:

   Preorder; the operator is printed before the operands.

   ii. What is the result of evaluating the expression?

   Answer:
   The string must represent the tree

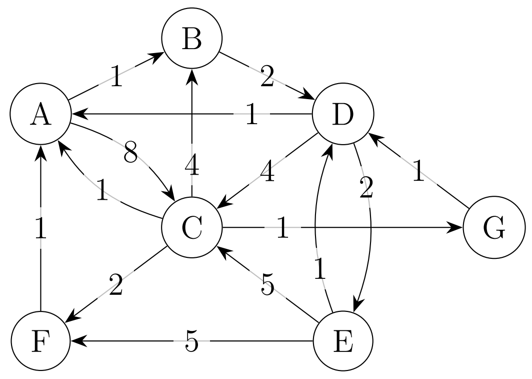


   This expression represents $(3 + 5) + (4 \cdot 2)$, *which is* 16.

   (d) [7 pts] Let $f(n) ::= n^2 + n \log n$. Then $f(n)$ is $O(g(n))$ for which of the following g? Note: there may be more than one.

- $g(n) ::= n \log n$ **False**
- $g(n) ::= n^2$ **True**
- $g(n) ::= 2^n$ **True**
- $g(n) ::= \log n$ **False**
- $g(n) ::= (n^2 + n \log n)/2$ **True**
- $g(n) ::= n^2 + n \log n - 1$ **True**
- $g(n) ::= \sqrt{n^2 + n \log n}$ **False**

2. Graphs [18 pts] Consider the following graph:



(a) [5 pts] List the nodes in the order they would be visited by DFS, starting from A. When there are multiple valid options, list the lower-letter nodes first. (e.g. if you can visit either C or F next, visit C next)

Answer:

A B D ~~*C F C E*~~ E C F G

(b) [5 pts] List the nodes in the order they would be visited by BFS, starting from A.

Answer:

A B C D F G E

(c) [5 pts] List the nodes in the order they would be visited by Dijkstra's algorithm, starting from A. Write down the resulting path lengths.

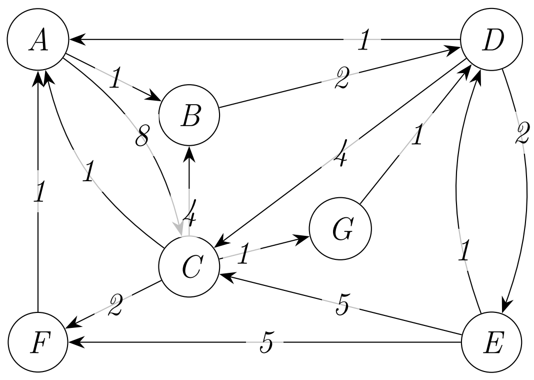
[TODO — clarification: "settled" instead of "visited"]

Answer:

A	*B*	*D*	*E*	*C*	*G*	*F*
0	*1*	*3*	*5*	*7*	*8*	*9*

(d) [3 pts] Is the graph planar?

Answer:
Yes.



3. Complexity [16 pts]

(a) [4 pts] What is the definition of "$f(n)$ is $O(g(n))$"?

Answer:
There exist constants $c > 0$ and $N > 0$ such that for all $n \geq N$, $f(n) \leq cg(n)$.

(b) [6 pts] Prove that $2n + 2$ is $O(n^2 - 1)$.

Answer:
Let $N = 2$ and let $c = 2$. Then if $n \geq N$, then $n^2 > 2n$. We have

$$\begin{aligned} 2n + 2 &\leq n^2 + 2 && \textit{since } n^2 \geq 2n \\ &\leq n^2 + 2 + (n^2 - 4) && \textit{since } n^2 \geq N^2 = 4 \\ &= 2n^2 - 2 = c(n^2 - 1) \end{aligned}$$

as required.

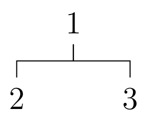
(c) [6 pts] Give the worst-case and expected case run times of quicksort on an array b of size n. Describe a situation in which it will take the worst-case time.

Answer:
The expected running time is $O(n \log n)$. The worst-case time is $O(n^2)$; this could occur, for example, if b is sorted. In this case, partitioning an array of size k will always split the array into a subarray of size 0 and a subarray of size $k - 1$, so each recursive call to quicksort will be on an input of size $k - 1$; leading to $O(n^2)$ total time.

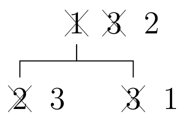
4. Heaps [14 pts]

In this question, you will perform operations on a min-heap in which the values are the priorities. When you change the value of a node, cross out the old value and write the new value next to it. For example, after swapping the root with the left child and then swapping the root with the right child,

[TODO — there is a typo in the example; should be right then left]

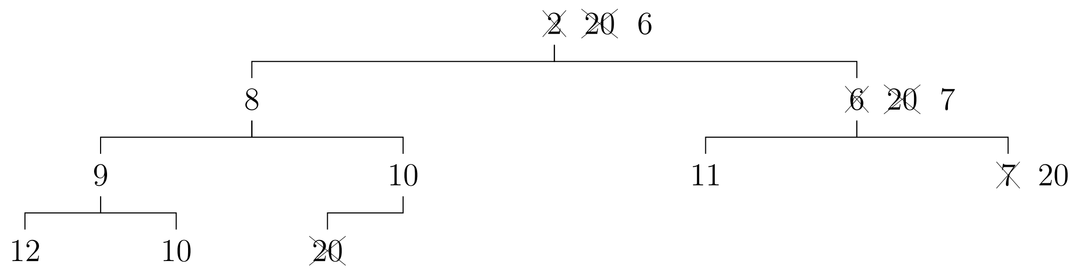


becomes

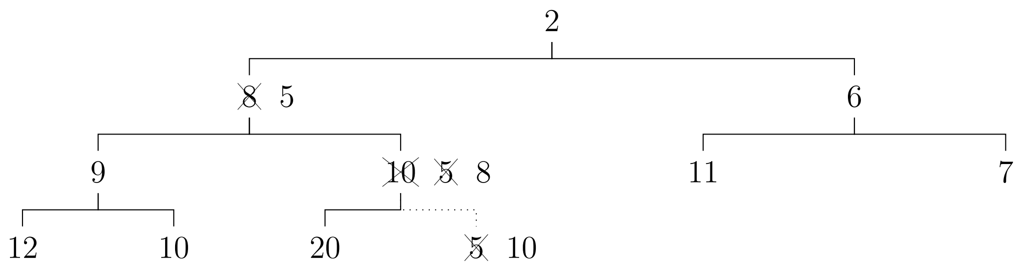


If you make a mistake, be sure that you differentiate between the mistake and the values that you are crossing out!

(a) [7 pts] Perform poll on the following min-heap:



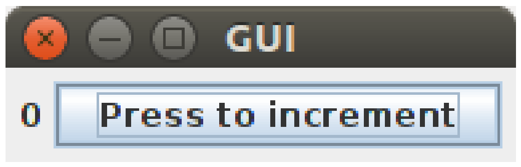
(b) [7 pts] Perform add(5) on the following min-heap:



5. GUIs [10 pts] Consider the program on the next page.

(a) [5 pts] Draw the GUI resulting from running the program on the following page.

Answer:



(b) [5 pts] The code doesn't work: nothing happens when the button is pressed. Explain why not, and fix it. Hint: this requires a one-line change. Don't worry if you don't remember the exact names of any swing methods you may need.

Answer:
No action listener is ever added to the button. The code has been modified so that the constructor of IncrementButton adds itself as an action listener on itself.

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

class GUISol extends JFrame {

  /** invariant: label.getText() = count */
  private JLabel label;
  private int    count;

  private class IncrementButton
        extends JButton
     implements ActionListener
  {
    public IncrementButton() {
      super("Press to increment"); addActionListener(this);
    }

    /** increment count and update label. */
    public void actionPerformed(ActionEvent e) {
      count = count + 1;
      label.setText("" + count);
    }
  }

  public GUISol() {
    super("GUISol");

    this.label = new JLabel("0");
    this.count = 0;

    setLayout(new FlowLayout());
    add(label);
    add(new IncrementButton());
    pack();
  }

  public static void main(String[] args) {
    new GUISol().setVisible(true);
  }
}
```

6. Spanning Trees [10 pts] Recall the following definition:

   Definition: A spanning tree T of a connected graph G is a subgraph of G containing all the nodes of G and a maximal set of edges, that

   (a) contains no cycle, and
   (b) connects all the nodes of G.

   Based on this definition, describe a high-level algorithm (as done in lecture) to find a spanning tree of G.

   Answer:
   One possible answer: Create a subgraph G' of G containing all nodes of G and no edges. While G' is not connected, choose an edge of G that does not form a cycle and add it to G'.

   Another possible answer: Repeat until no longer possible: Find a cycle in G and delete one edge of the cycle.

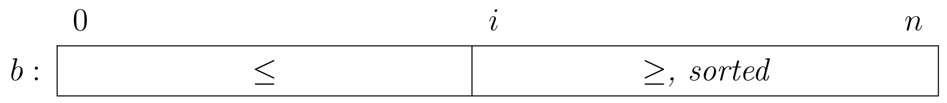
   Any correct algorithm is acceptable.

7. Sorting [14 pts] In this question we develop a sorting algorithm using loop invariants.

   - Precondition: the array b has length $n+1$ (nothing is known about the values).
   - Postcondition: $b[0..n]$ is sorted.
   - Invariant: $b[i..n]$ is sorted and $b[0..i-1] \le b[i..n]$.

   (a) [2 pts] Give an array diagram that captures this invariant.

   Answer:



   (b) [2 pts] Give the initialization statements that make the invariant true.

   Answer:
   i = n+1;

   (c) [2 pts] Give the loop condition.

   Answer:
   i != 0; or i > 0

   (d) [2 pts] Give a statement that makes progress towards termination (it may violate the invariant).

   Answer:
   i = i − 1;

(e) [3 pts] Describe in English what must be done in the loop body before part (d) to preserve the invariant. Your answer should be roughly one sentence.

Answer:

Find the largest element in $b[0..i-1]$ *and swap it with* $b[i-1]$.

(f) [3 pts] What is the asymptotic worst-case running time of this algorithm?

Answer:

$O(n^2)$