Prelim 2

CS2110 Spring 2014

April 21, 2014

	1	2	4	5	Total
Question	True False	Short Questions	Minimax & Trees	Graphs	Total
Max	16	34	25	25	100
Score					
Grader					

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 the top of *every* page! There are 4 questions on 9 numbered pages. Check now that you have all the pages. When you hand in your exam, make sure your booklet is still stapled together. If not, please use our stapler to reattach all your pages.

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. Use the back of pages if necessary; just make sure to make it clear where the answers are.

Ambiguous answers will be considered incorrect.

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.

1. True / False (16 points)

a)	T	F	If a method contains an if-statement "`if (...) int x= 5; ...`", space for local variable x is allocated every time the if-condition is found true and then deallocated when the if-statement is finished.
b)	T	F	The following function declared in class C overrides function `equals` in class Object: `public boolean equals(C ob) { return this == ob; }`
c)	T	F	Consider the declaration "`class C extends B implements C1, C2 {...}`" and C contains no constructor. The following expression is legal and will not cause a runtime error: "`(C1)(C1)(B)(new C())`" .
d)	T	F	Suppose class C contains an inner class IC with a private field `x`. Methods declared in class C can reference field `x`.
e)	T	F	Consider the declaration "`public class C extends B {...}`". Somewhere, we declare variable v using "`ArrayList<B> v= new ArrayList<B>;`" Then the call `v.add(new C());` is not syntactically legal because a C is not a B.
f)	T	F	Suppose classes `Dog` and `Cat` extend class `Animal`. Only class Dog declares method `bark()`. Let `kitty` be an instance of `Cat`. `((Dog)(Animal) kitty).bark()` will upcast `kitty` to `Animal`, downcast to `Dog`, and then run its `bark()` method.
g)	T	F	Suppose that for two objects `b` and `c` of class `C`, `b.equals(c)` is false. Then the specifications of `equals` and `hashCode` in the Java API package require that `b.hashCode() != c.hashCode()`.
h)	T	F	If a GUI includes a JFrame, you can add a large number of components to that JFrame with the default layout manager for JFrame.
i)	T	F	Your class `C` implements interface `ActionListener` and thus implements procedure `actionPerformed(ActionEvent e)`. That's all you have to do in order to have `actionPerformed` called when a `JButton j` on the GUI is clicked.
j)	T	F	Let s be a String variable. The call `s.indexOf(r)` always takes time `O(s.length())` in the worst case, regardless of the length of `String r`.
k)	T	F	The worst case runtime complexity of Binary Search is $O(n \log n)$.
l)	T	F	After inserting n elements into a Binary Search Tree, the BST will always have depth $\log_2 n$.
m)	T	F	Heapsort uses only $O(1)$ space, so it is more space-efficient than mergesort.
n)	T	F	Selection sort always makes $O(n)$ swaps and $O(n^2)$ array comparisons to sort an array of n elements.
o)	T	F	Let $f(x) = x^2 + 500$, and let $g(x) = 100 * x^2$. Then $g(x)$ is $O(f(x))$.
p)	T	F	Using $f(x)$ and $g(x)$ as defined in the previous question, $f(x)$ is $O(g(x))$.

2. Short Questions [34 points]

2.a Hashing [5 points]

Consider hashing using linear probing to implement a set, as discussed in recitation. Assume an array of 5 of elements of class Integer, as shown below. Let the hash function be the square of the value modulo the array size, e.g. hashing 5 gives 0 (which is 25 mod 5).

```
  ----------------
0 |_null___________|
1 |_null___________|
2 |_null___________|
3 |_null___________|
4 |_null___________|
```

(1) Try to insert the values 1, 2, 2, 3, 4, 5 into the set, in that order —write the values in the appropriate element in the table above, crossing off the value currently in that element. But do not change the size of the array, even though the technique calls for it.

(2) Now remove the value 4 from the set. Do you simply set the array element to `null`? Explain why or why not.
Answer.

2.b Heaps. [5 points]

Consider maintaining a heap in an array h[0..]. Write the body of the following function.

```
/** Return the index of the parent of node h[k], i.e. whose index is k.
  *  Precondition:  k > 0. */
public static int parentOf(int k) {




}
```

2.c Game-tree and Mini-max. [6 points]

(1) In Connect 4, starting from the first move, the possible number of game states are (choose the closest option):
A. $2^{7\times 6}$ B. $2^7 \times 2^6$ C. $(6 \times 7)^2$ D. 6×7.

(2) In Connect 4, with a finite horizon, i.e. only evaluating the AI for a finite depth, will an AI of a larger depth win over an AI of smaller depth?
A. Yes, always. B. Yes, most of the times. C. No

2.d Sorting/Searching. [7 points]

(1) Below, write the body of procedure QS. You may use English in appropriate places, and you may assume that an appropriate function does the partition algorithm, as we did in lecture.

```
/** Sort b[h..k], using the quicksort algorithm. */
public static void QS(int[] b, int h, int k) {









}
```

(2) How do you change Quicksort so that it takes $O(\log n)$ instead of $O(n)$ space in the worst case? You do not have to write the code; just tell us the idea; it can be explained in one sentence.
Explain briefly:

2.e Induction. [5 points]

Prove by induction: A complete balanced tree of depth n has 2^n leaves. (The depth is the length of the longest path from the root to a leaf; by "complete" we mean it has as many leaves as possible.) First state the theorem in terms of a property $P(n)$, as done throughout the lecture on induction.
Proof.

2.f Exception handling. [6 points]

(1) Consider the statement below, appearing in a method `m`. Does its execution result in a `RuntimeException` being thrown out to the call of `m`? Write your answer and an explanation for it to the right of the statement.

```
try {
    return b/0;
}
catch (RuntimeException r) {
    return b/0;
}
```

(2) To the right below, write down what is printed by the `println` statements during execution of the call `mm(0)`, where method `mm()` is defined as follows:

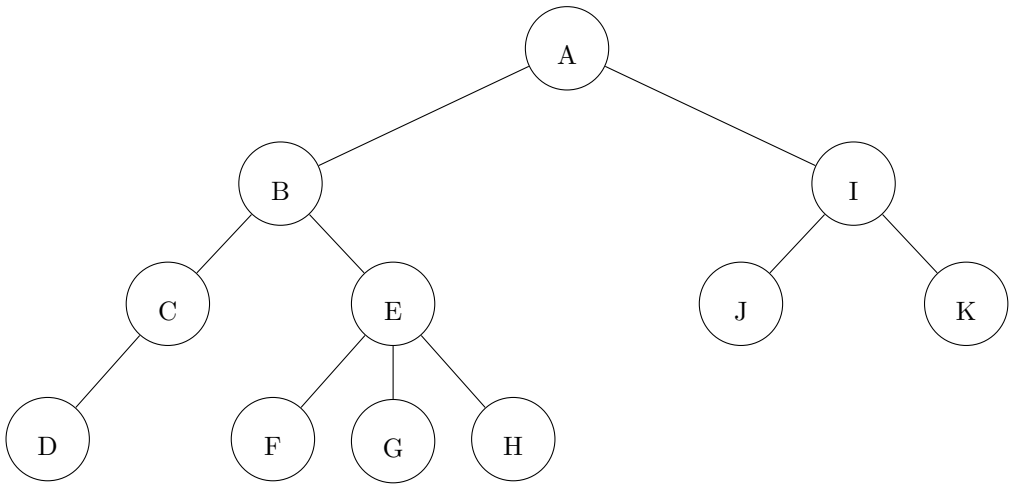
```
public static int mm(int x) {
    try {
       System.out.println("one");
       int b= 5/x;
       System.out.println("two");
       return b;
    } catch (RuntimeException e) {
       System.out.println("three");
       int c= 5/(x+1);
       System.out.println("four");
    }
    System.out.println("five");
    int d= 5/x;
    System.out.println("six");
    return d;
}
```

3. Minimax Game Tree [25 points]

(a) [5 points] Is the Minimax algorithm, as you implemented it in Assignment 4, closer to DFS or BFS in the way it traverses the game tree? Why?

(b) [10 points] Consider the following game, which is a variant of Nim: There are 21 sticks in a pile. Two players take turns removing sticks from the pile, and the winner is the player to take the last stick. Each player can take up to 3 sticks from the pile on their turn. Show the `depth-2` game tree (i.e. a total of three levels) that an AI using the Minimax algorithm would construct for this game, if there are 10 sticks left in the pile when it is the AI's turn. Be sure to label each node of the tree with the state of the board (i.e. the pile) at that node.

(c) [10 points] The following game tree has been constructed by the AI player in a game. The details of the game are not important, and the state of the board at each node is represented by a capital letter. The table below the tree contains the values of function `evaluateBoard()` for all board states. Use the Minimax algorithm to label each node with its value, then circle the child indicating the move the AI should make to maximize its expected score.



Board state	A	B	C	D	E	F	G	H	I	J	K
`evaluateBoard()`	5	6	1	2	-2	-6	-1	0	3	4	-5

4. Graphs [25 points]

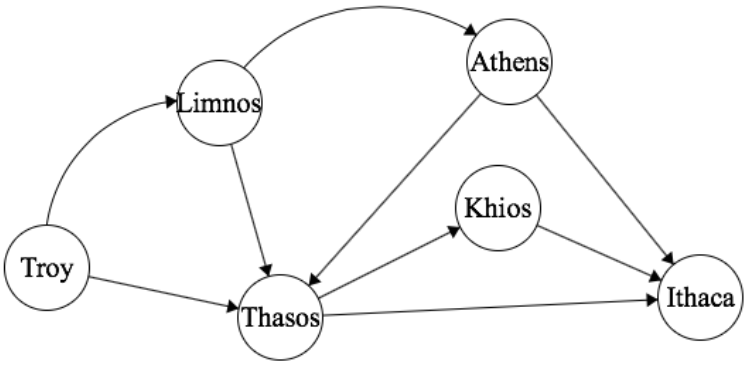


Figure 1: The possible routes from Troy to Ithaca

(1) [4 points] Write a topological sort ordering for the graph above. That is, give numbers 0 to 5 for each of the nodes above.

(2) [4 points] Odysseus, currently in the city of Troy, wants to search the cities in the above graph for his missing friend Telemachus. He knows that Telemachus is waiting for him in one of the cities. So, Odysseus, being in a rush, reasons that he will visit each city at most once. Choose an algorithm that can be used to navigate the graph efficiently to find Telemachus. Name the algorithm you chose and briefly justify your choice.

(3) [12 points] It turns out that each route has a cost associated with it. In the graph on the next page, the number on an edge from one city to another is the number of days it takes to get from the one city to the other.

Using Dijkstra's algorithm, compute the shortest path from Troy to Ithaca in terms of the time it takes to get from one city to another, writing information in the table below. The column labeled "before" shows the initial path-value for each city just before the loop of the algorithm is executed. Each column represents an iteration of the loop of the algorithm, the first one being iteration 0. For each iteration, write in its column the value for the city whose value changes at that iteration. If a value doesn't change at that iteration, don't write anything in that cell.

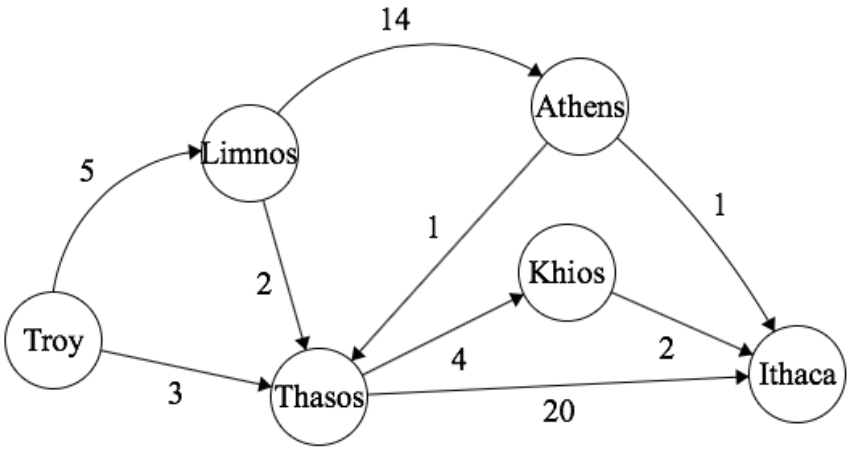


Figure 2: The possible routes from Troy to Ithaca

	before	0	1	2	3	4	5
Troy	0						
Limnos	∞						
Thasos	∞						
Athens	∞						
Khios	∞						
Ithaca	∞						

(4) [5 points] Suppose that the graph extends past the 6 cities as shown in the graph below. (Each node labeled "..." can also be connected to nodes that we can't see.) The world past these cities is unexplored, meaning that there could potentially be an infinite number of cities in the extended graph. Odysseus wants to find the city of Thasos. He knows that it's only a couple edges away from Troy. Does it make more sense to use Breadth-First Search or Depth-First Search to find Thasos on this potentially infinite graph? Explain your answer.