CS61B, Fall 2015 Final Examination (with corrections) P. N. Hilfinger

READ THIS PAGE FIRST. Your exam should contain 14 problems on 16 pages. Officially, it is worth 46 points.

This is an open-book test. You have three hours to complete it. You may consult any books, notes, or other inanimate objects available to you. You may use any program text supplied in lectures, problem sets, or solutions. Please write your answers in the spaces provided in the test. Make sure to put your name, login, and lab section in the space provided below. Put your login and initials *clearly* on each page of this test and on any additional sheets of paper you use for your answers.

Be warned: our tests are known to cause panic. Fortunately, this reputation is entirely unjustified. Just read all the questions carefully to begin with, and first try to answer those parts about which you feel most confident. Do not be alarmed if some of the answers are obvious. Should you feel an attack of anxiety coming on, feel free to jump up and run around the building once or twice.

Your name: ________________________ Login: __________

Login of person to your Left: ______________ Right: ______________

Discussion TA: ______________________________________

1. ____/2 2. ____/4 3. ____/ 4. ____/6 5. ____/4

6. ____/4 7. ____/4 8. ____/4 9. ____/4 10. ____/4

11. ____/2 12. ____/2 13. ____/2 14. ____/4

Please sign the following:

> I pledge my honor that during this examination I have neither given nor received assistance.

Signature: ____

1. [2 points]

a. Describe as concisely as possible what `foo` does for all but one non-negative value of `x`.

```
public static int foo(int x) {
    int y = 1;
    x = x ^ y;
    while ((x & y) == 0) {
        y = y << 1;
        x = x ^ y;
    }
    return x;
}
```

b. What is the remaining value of `x` and what number results from applying `foo` to it?

2. [4 points] Consider a max heap of integers that is stored in an array, with the root vertex at index 1 (and the element at index 0 unused). Define the length of the edge between a vertex and a child of that vertex as the value of the parent minus that of the child. Fill in the function below to fulfill its comment. Use only one expression or statement per blank.

```
/** Given that A is a max heap with root at index 1, and U>=1 and V>=1
 *  are valid indices of vertices in the heap, returns the length of the
 *  shortest path from U to V, where the length of an edge is the
 *  value at the parent vertex minus the value at the child. */
public static int shortestHeapPath(int[] a, int u, int v) {
    int u1 = u; v1 = v;

    while (___________________________) {

        if (___________________________) {

            ___________________________;
        } else {

            ___________________________;
        }
    }

    return ________________________________________;
}
```

3. [1 point] What do the words “murder,” “skulk,” “parliament,” and “exaltation” have in common?

4. [6 points] In the questions below, give the bounds asked for in asymptotic ($\Theta(\cdot)$, $O(\cdot)$) notation, as a function of the value of N in each case. Give your answer in as simple a form as you can (e.g. $\Theta(N^9)$, not $\Theta(12N^9)$), and give as tight a bound as possible.

a. Assume that `L` is a linked list, N is its length, and calls to `test` take constant time. What is the worst-case running time for executing:

```
IntList q = L;
int c; c = 0;
for (IntList p = L; p != null; p = p.tail) {
    while (q != null && test(q, p)) {
        c += 1;
        q = q.tail;
    }

}
```

Answer: ____________

b. Assume that `Vertex` is the vertex type for a connected, directed, acyclic graph in which each vertex has a maximum out-degree of 3. Define

```
boolean find(Vertex v, int x) {
    for (Vertex s : v.successors()) {
       if (s.getLabel() == x || find(s, x)) {
            return true;
       }
    }
    return false;
}
```

Let N be the number of edges in the graph. For `Vertex` w, find a bound for the worst-case running time of

```
find(w, 42);
```

Answer: $O($____________$)$

c. Repeat part (b), but assume that the graph is a tree. **Answer:** ____________

Continues on next page.

d. Give an asymptotic bound on

$$\sum_{0 \le i \le k} (\lg N)^i N^{k-i}$$

Answer: $\Theta($____________$)$

e. True or false: If $f(N) \in O(N^2)$ and $g(N) \in O(N)$ are positive-valued functions, then $f(N)/g(N) \in O(N)$. If true, explain why; if false give a counterexample.

f. True or false: If $f(N) \in \Theta(N^2)$ and $g(N) \in \Theta(N)$ are positive-valued functions, then $f(N)/g(N) \in \Theta(N)$. If true, explain why; if false give a counterexample.

5. [4 points] Consider the following definition of a tree type with integer labels:

```
public class IntTree {
    public int value;
    public IntTree left, right;
    // Perhaps other members.
}
```

Fill in the method `closestLabel` to fulfill its comment. For a "bushy" argument `T`, its running time should be $O(\lg N)$. To simplify coding, feel free to use the methods `closer`, below.

```
/** Return whichever of Y or Z is closer to X. */
static int closer(int x, int y, int z) {
    return Math.abs(x - y) < Math.abs(x - z) ? y : z;
}

/** Assuming T is a BST, return the label in T that is closest to
 *  X, or DFLT if T is empty. */
static int closestLabel(IntTree T, int x, int dflt) {
    if (T == null) {
       return dflt;
    }


    if (_________________________) {

        return ________________________________________________________;

    else if (_________________________) {

        return ________________________________________________________;
    } else {

        return ________________________________________________________;
    }
}
```

6. [4 points] Consider an $N \times N$ grid whose cells contain integer values. We wish to find a shortest path from the lower-left corner of the grid to the upper-right corner, where a path consists of a sequence of squares, each one of which is adjacent to the preceding square in the path and either above, to the right, or diagonally above and to the right of the preceding square. The length of the path is the sum of the integers in its squares. For example, the shaded path shown below is shortest:

3	6	3	2
5	1	8	10
7	4	2	1
2	3	1	1

Here's a pseudo-code function to find the length of the shortest path from row i, column j to row N, column N, for $1 \le i \le N$, $1 \le j \le N$, in an $N \times N$ grid G. $G(r, c)$ is the value at row r, column c of G (numbering from (0, 0) in the bottom left corner. Unfortunately, its running time of `lsp` is exponential.

```
def lsp(i, j):
    if i >= N or j >= N:
        return infinity
    if i == N-1 and j == N-1:
        return G(i, j)
    else:
        return G(i, j) + min(lsp(i+1, j), lsp(i, j+1), lsp(i+1, j+1))
```

a. Fill in the program below to compute the result in polynomial time into `L[0][0]`. You may assume you have a multi-argument `min` function available, as in Python.

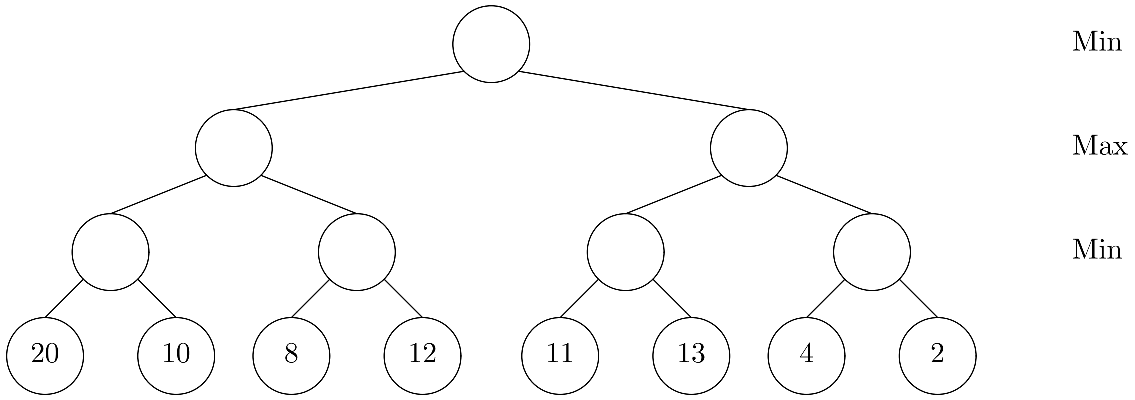
```
int[][] L = new int[N][N];
L[N-1][N-1] = G(N-1, N-1)

for (int c = __________; c __________; c__________) {
    __________
}

for (int r = __________; r __________; r__________) {
    __________ = __________;
    for (int c = __________; c __________; c__________) {
        ______________________________________
    }
}
```

b. As a function of N, what is the worst-case execution time of this program?

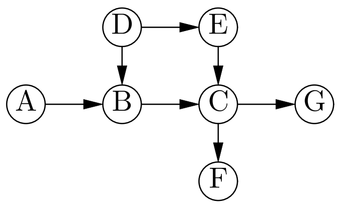
7. [4 points] Given the following minimax tree (where the root node is a *minimizing* node):



a. What is the value of the root node?

b. Circle the nodes that will not be examined as a result of alpha-beta pruning (assuming we process child nodes left-to-right).

8. [4 points] Consider the following directed graph:



a. Give a valid topological ordering of its nodes.

b. Is it possible to add an edge to the graph so that no valid topological sort exists? If so, indicate such an edge. Otherwise, briefly indicate why no such edge exists.

Continues on the next page

c. Consider the following pseudo-code for operating on a directed graph, G:

```
1. Let $G'$ be a copy of $G$, with all edges reversed.
2. Initially, all vertices of $G'$ are colored white and are unnumbered.
3. Set $n$ to 0.
4. For each vertex, $v$, of $G'$,
   4.1 If $v$ is unnumbered, execute tsort($v$).
```

where the function tsort is defined

```
5. tsort($w$):
   5.1 Color $w$ gray.
   5.2 For each unnumbered successor, $s$, of $w$ in $G'$,
       5.2.1 tsort($s$)
   5.3 Give $w$ the number $n$.
   5.4 Increment $n$.
```

When this algorithm finishes execution on a graph that has a valid topological sort, what is a correct order for sorting the vertices of G?

d. What test can you add to the function tsort to raise an error if G has no valid topological sort?

9. [4 points] For each of the following, give a data structure or algorithm that you could use to solve the problem or write "impossible" if it is impossible to meet the running time given in the question. For each question, we have one or more right answers in mind, all of which are among the data structures and algorithms listed below. For each answer, provide a *brief* description of how the algorithm or data structure can be used to solve the problem.

Possible answers (you can use some more than once and not use others at all):

DFS	Heap	Quick sort	Impossible
BFS	Trie	Merge sort	
Dijkstra's	Hash table	Insertion sort	
Topological Sort	Balanced BST	Radix sort	
Kruskal's	Linked list	Selection Sort	

a. Given a list of N words with k characters each, find for each word its longest prefix that is also the prefix of some other word in the list. Worst-case running time: $O(Nk)$ character comparisons.

b. Given an undirected weighted connected graph with $|E|$ edges, find the heaviest edge that can be removed without disconnecting the graph. Worst-case running time: $O(|E|log|E|)$.

c. We would like to build a data structure that supports the following operations: add, remove, and find the k^{th} largest element for any k. Worst-case running time: $O(log(N))$ comparisons for each operation (where N is the number of elements currently in the data structure).

d. Given an unordered list of N Comparables, construct a BST containing all of them. Running time: $O(N)$ comparisons.

10. [4 points]

a. Draw the graph corresponding to the following adjacency matrix (vertex labels A–D for the rows and columns are above and to the left:

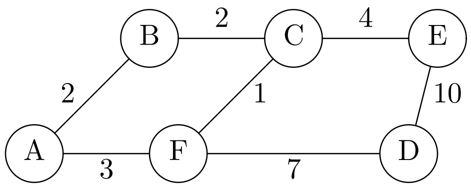
$$\begin{array}{c|cccc|}
 & \text{A} & \text{B} & \text{C} & \text{D} \\
\hline
\text{A} & 0 & 1 & 0 & 0 \\
\text{B} & 0 & 1 & 0 & 1 \\
\text{C} & 1 & 0 & 0 & 1 \\
\text{D} & 0 & 0 & 0 & 0 \\
\hline
\end{array}$$

b. Give the adjacency-list representation of the above graph.

c. Using an adjacency-list representation of a graph, how long does it take in the worst case to check for the existence of an edge between two vertices? Answer in terms of $|V|$ and $|E|$, the number of vertices and edges in the graph, respectively, using $\Theta(\cdot)$ notation.

d. How long does the same operation take using an adjacency matrix, in the worst case?

11. [2 points] When conducting an A* search on the following graph, for what range of values can the heuristic function return in order to make the heuristic admissible? Our heuristic function h takes in two vertices, x, and y, and returns a number for the estimated shortest path length from x to y.



____________ $\leq h(B, D) \leq$ ____________, ____________ $\leq h(A, D) \leq$ ____________

12. [2 points] Consider an undirected, connected graph G where all edges have a weight of 1, and a spanning tree T of G. For any two distinct vertices s and t in G, define $L_T(s, t)$ to be the length of the path from s to t in T, and define $L_G(s, t)$ to be the length of the shortest path from s to t in G.

a. If N is the number of vertices in G, what is the maximum possible value of $L_T(s, t)/L_G(s, t)$ for any two vertices of G? Give an exact expression, without using asymptotic notation ($\Theta(\cdot)$, etc.).

Answer: ____________

b. Draw an example of a graph and an MST of that graph that achieves this ratio. Be sure to indicate the nodes s and t in your example.

13. [2 points] The (unmemoized) `hashCode` method for the class `java.lang.String` is as follows:

```
public int hashCode() {
    int h = 0;
    for (int i = 0; i < length() ; i++) {
        h = 31 * h + charAt(i);
    }
    return h;
}
```

For this problem, assume that a `HashSet<String>` uses the value of this function by taking it modulo the number of buckets, after first masking off the sign bit to make the number non-negative. (The actual `HashSet` and `HashMap` implementations do something more sophisticated.)

a. Given a `String` of length L and a `HashSet<String>` that contains N Strings, give the worst- and best-case running times of inserting the `String` into the `HashSet`.

   Worst case: $\Theta($______________$)$ String comparisons

   Best case: $\Theta($______________$)$ String comparisons

b. In Java, `HashSets` always use arrays whose size is a power of two. If this were not the case, the `hashCode` method shown above could be a very poor hash function. Give an example of an integer N such that `hashCode` would be a very poor choice of hash function for a `HashSet` whose array had size N. Give a brief explanation of your answer. Assume that the Java `HashSet` uses external chaining to resolve collisions.

14. [4 points] Complete the following program by filling in the blank line on this page and adding any necessary code to the next page.

```
public interface Tree {
    /** Return true iff I contain X. */
    boolean contains(int x);

    /** Insert X into me, if not already present, returning the result. */
    Tree insert(int x);

    /** The empty tree, containing nothing. [NOTE: static methods
     *  in interfaces is a Java 8 feature.] */
    static Tree emptyTree() {
        return _____________________________________;
    }
}

public final class RegularTree implements Tree {
    /* NOTE: Final classes cannot be extended. */
    /** A Tree containing label LAB and the contents of trees L and R,
     *  where all elements of L are < LAB and those of R are > LAB. */
    RegularTree(int lab, Tree L, Tree R) {
        _label = lab; _left = L; _right = R;
    }

    @Override
    public boolean contains(int x) {
        return x == _label ? true :
               x < _label  ? _left.contains(x) :
                             _right.contains(x);
    }

    @Override
    public Tree insert(int x) {
        if (x < _label) {
            _left = _left.insert(x);
        } else if (x > _label) {
            _right = _right.insert(x);
        }
        return this;
    }

    private int _label;   private Tree _left, _right;
}
```

Continued on next page

Add any additional code here. It **may not** contain any **if** statements, **while** statements, **switch** statements, conditional expressions, or **try** statements.

```
    // FILL IN, IF NEEDED.
```

Scratch space

Scratch space