# Final Exam (Spring 2013)

Name:

Grades Nickname:

CADE login:

## 1 True/False (2 pts each):

1. F Specular lighting is independent of viewing direction.
2. T Whether a number is prime can be determined in $O(\sqrt{N})$ time.
3. T Every affine transformation can be decomposed into two rotations and a non-uniform scale.
4. T A hash function can overflow and still be good.
5. F Depth-first search requires a priority queue.
6. T A simple path contains at most one cycle.
7. T Storing a graph as an adjacency list requires $O(|V| + |E|)$ storage.
8. F The final structure of an AVL tree is independent of insertion order.
9. T A graph is sparse if $|E| = O(|V|)$.
10. F The algorithm for topological sort takes $O(|E|)$ time.
11. T! Adam taught this course really well ☺.

## 2 Runtimes

Give the Big-O running times of the following algorithms (2 pts each):

12. Dijkstra's Algorithm

    E log E

13. Kruskal's Algorithm

    E log E

14. Insertion into a heap

    log n

15. Heapify

    n

16. Strongly Connected Components

    E

## 3 Balanced BSTs

17. Insert 13 into the AA tree below (5 pts).

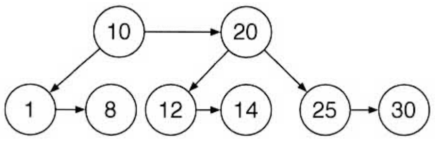
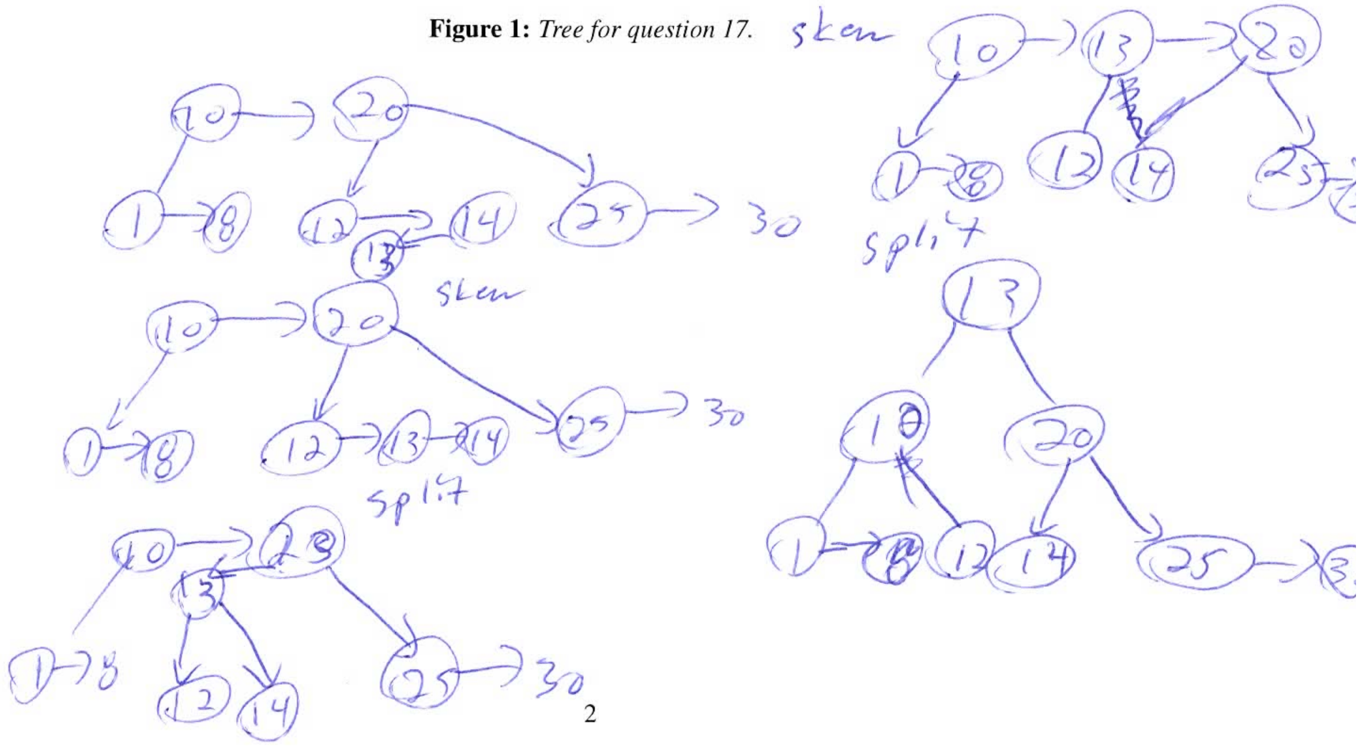


**Figure 1:** *Tree for question 17.*



18. Answer the following question about the tree in Figure 2 (5 pts each):

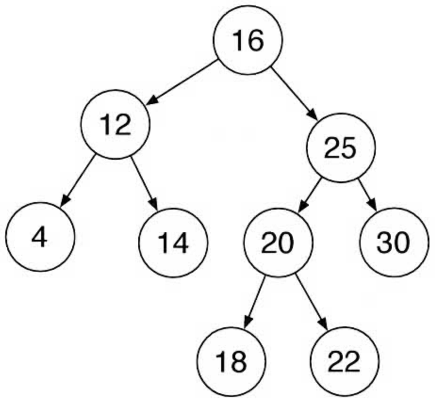
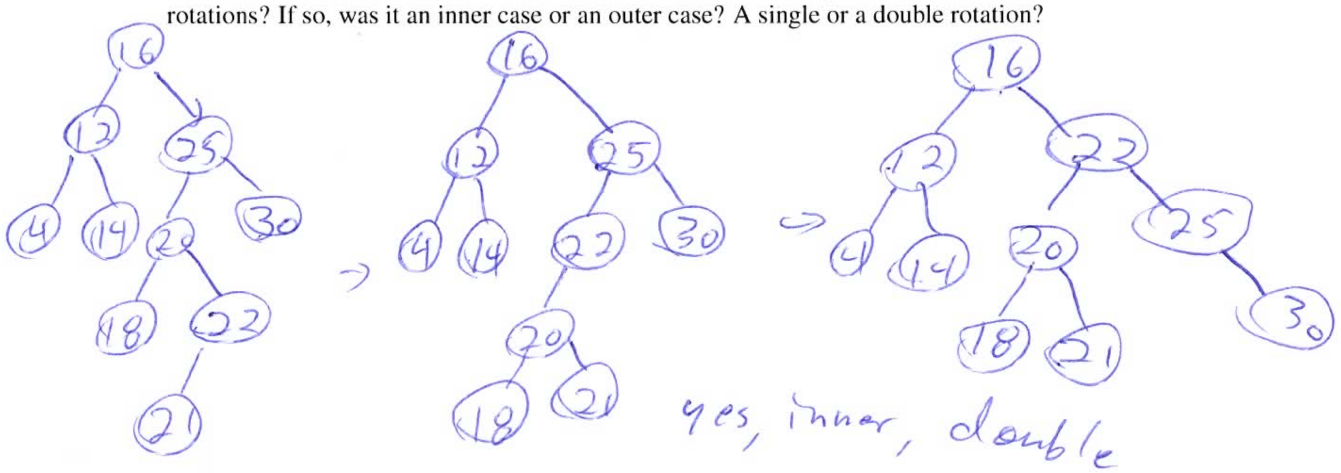


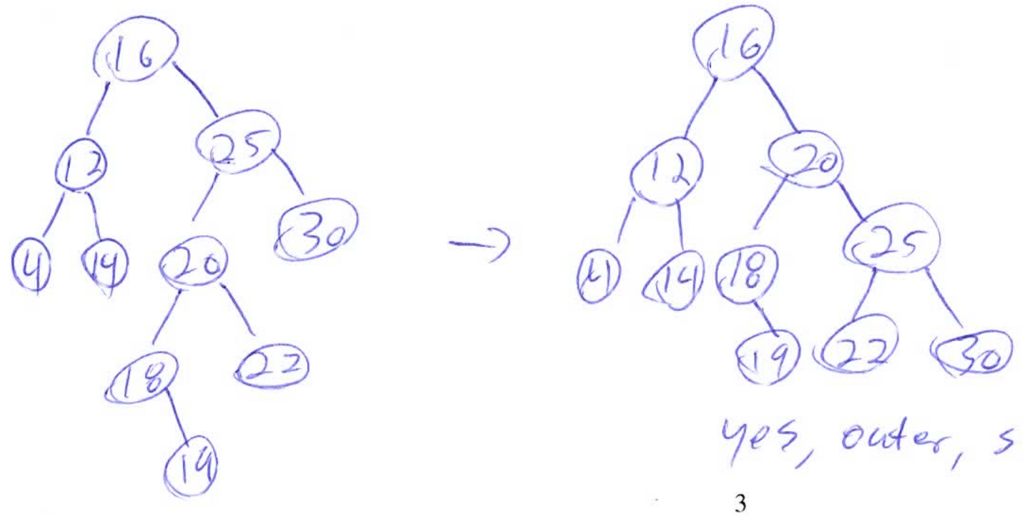
**Figure 2:** *Tree for question 18*

- Assume the tree is an AVL tree. Show the result of inserting 21 into the tree. Did this result in any rotations? If so, was it an inner case or an outer case? A single or a double rotation?



yes, inner, double

- Assume the tree is an AVL tree. Show the result of inserting 19 into the (original) tree. Did this result in any rotations? If so, was it an inner case or an outer case? A single or a double rotation?



yes, outer, single

## 4 KD-Trees

19. For the points in Figure 3:

    - Build a balanced kd-tree (nodes and edges) for the points in Figure 3 (5 pts).
    - Draw the separating planes in the figure (3 pts).
    - Highlight the tree edges traversed when looking for the point with minimum x-coordinate (2 pts).
    - Make another copy of the tree and highlight the edges traversed when looking for points within 3 units of $(16, 6)$ (2 pts).

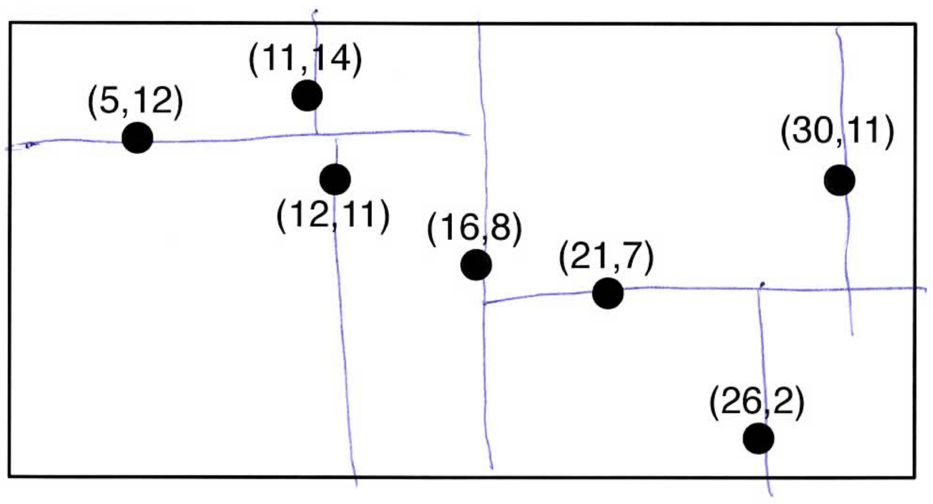
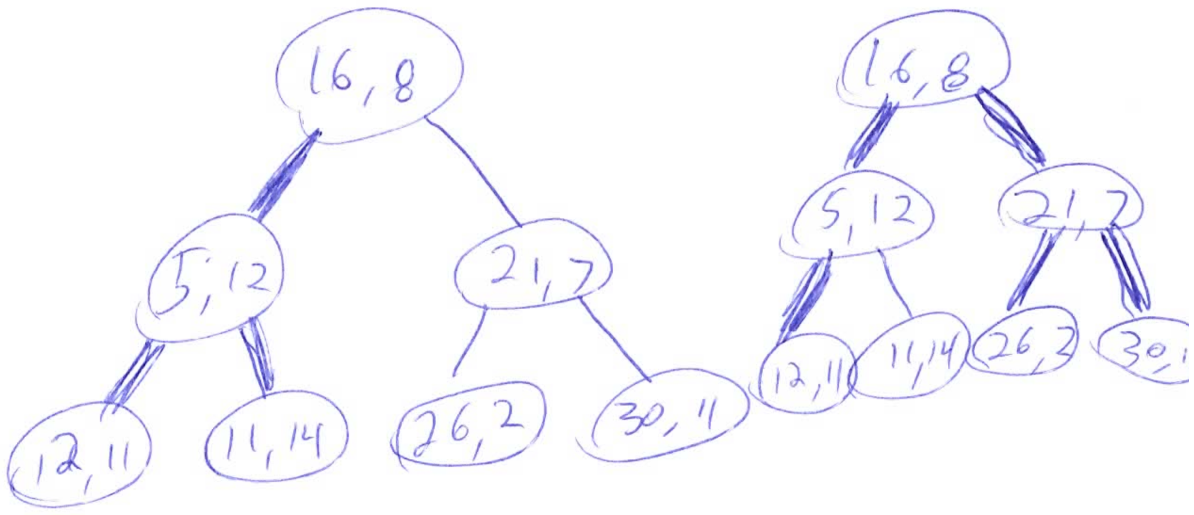


**Figure 3:** *Points for question 19*



## 5 Heaps

20. For the heap in Figure 4 show:

- (a) How the heap is stored in an array (4 pts)
- (b) the array resulting from inserting 8 into the heap from part (a) (5 pts)
- (c) the array resulting from removing the minimum from the heap in part (b) (5 pts).

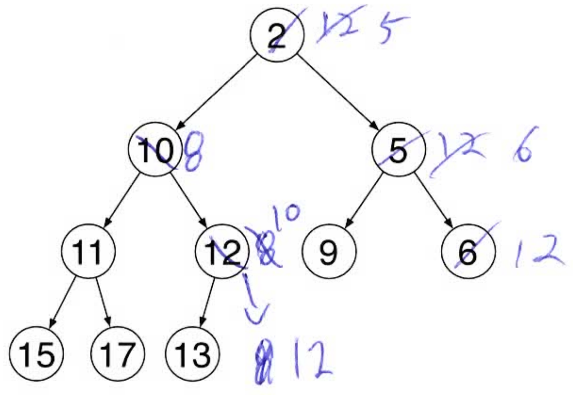


**Figure 4:** *Heap for question 20*

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| a) | 2 | 10 | 5 | 11 | 12 | 9 | 6 | 15 | 17 | 13 | | | | | |
| b) | 2 | 8 | 5 | 11 | 10 | 9 | 6 | 15 | 17 | 13 | 12 | | | | |
| c) | 5 | 8 | 6 | 11 | 10 | 9 | 12 | 15 | 17 | 13 | | | | | |

21. Run the linear time heapify/build-heap algorithm on the tree in Figure 5 (5pts).

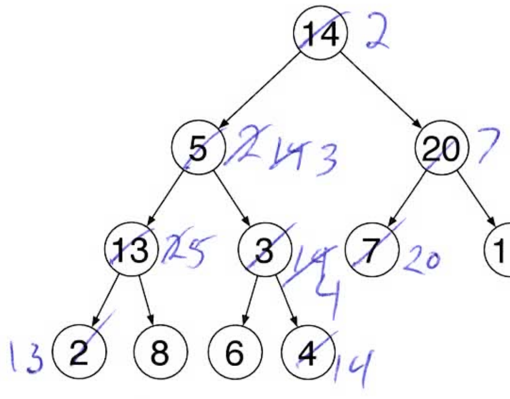
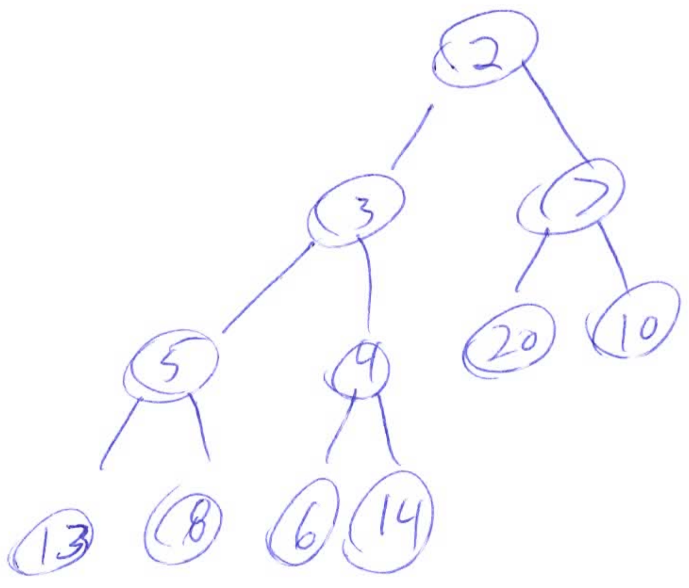


**Figure 5:** *Heap for question 21*



## 6 Hash Tables

22. Show the result of inserting the sequence $\{32, 18, 46, 19, 21, 6\}$ into a hash table of size 13 using (a) linear probing, (b) quadratic probing, and (c) separate chaining (3 pts each). Note: $32\%13 = 6$, $18\%13 = 5$, $46\%13 = 7$, $19\%13 = 6$, $21\%13 = 8$, $6\%13 = 6$.

| | |
|---|---|
| 0 | |
| 1 | |
| 2 | |
| 3 | |
| 4 | |
| 5 | 18 |
| 6 | 32 |
| 7 | 46 |
| 8 | 19 |
| 9 | 21 |
| 10 | 6 |
| 11 | |
| 12 | |

| | |
|---|---|
| 0 | |
| 1 | |
| 2 | 6 |
| 3 | |
| 4 | |
| 5 | 18 |
| 6 | 32 |
| 7 | 46 |
| 8 | 21 |
| 9 | |
| 10 | 19 |
| 11 | |
| 12 | |

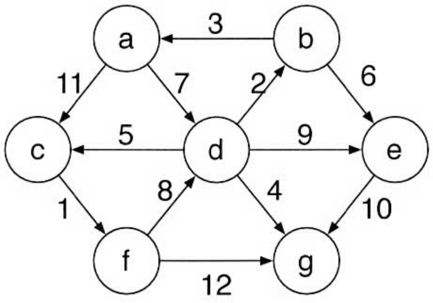
| | |
|---|---|
| 0 | |
| 1 | |
| 2 | |
| 3 | |
| 4 | |
| 5 | 18 |
| 6 | $32 \rightarrow 19 \rightarrow 6$ |
| 7 | 46 |
| 8 | 21 |
| 9 | |
| 10 | |
| 11 | |
| 12 | |

What is the load factor of the hash table after the insertions (2 pts)?

6/13

## 7 Graphs

23. Give the order of vertices visited in (a) Breadth First Search starting at node $c$ (b) Depth First Search starting at node $b$ and (c) Dijkstra's Algorithm when starting at node $a$ in the graph in Figure 6. Break ties using alphabetical order (i.e. if the algorithm could go to node $c$ or $e$, visit $c$ first). Show the state of any auxiliary data structure (i.e. stack, queue, or priority queue) after each node is first visited. (5 pts each).



$$\begin{tikzcd}
& a \arrow[dl, "11"'] \arrow[dr, "7"] & & b \arrow[ll, "3"'] \arrow[dr, "6"] & \\
c \arrow[dr, "1"'] & & d \arrow[ll, "5"'] \arrow[ur, "2"] \arrow[rr, "9"] \arrow[dr, "4"] & & e \arrow[dl, "10"] \\
& f \arrow[ur, "8"] \arrow[rr, "12"'] & & g &
\end{tikzcd}$$

**Figure 6:** *Graph for question 23*

- BFS($c$): ~~c, f, d, b, a, e, g~~ c, f, d, g, b, e, a
- DFS($b$): b, a, c, f, d, e, g
- Disjkstra($a$): a 0, d 7, b 9, c 11, [illegible] g 11, f 12, e 16

d 7  c 11 [illegible] c 12 [illegible]  b 9  e 16  g 11  g 12  e 17

f 12  g 24  d 20

24. Run the topological sort algorithm on the graph in Figure 7 (5 pts).

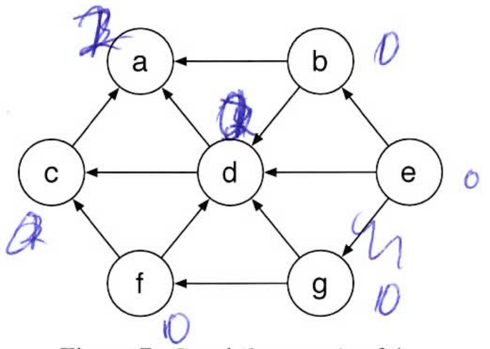


**Figure 7:** *Graph for question 24*

e, b, g, f, d, c, a

25. Run the strongly connected components algorithm on the graph in Figure 8. Show your work by recording the order that nodes are visited. (5 pts).

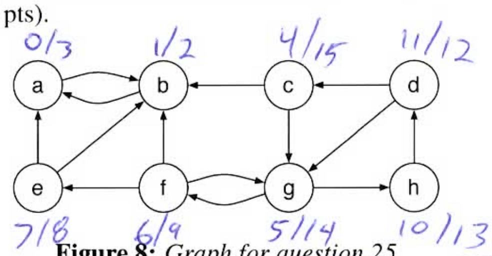
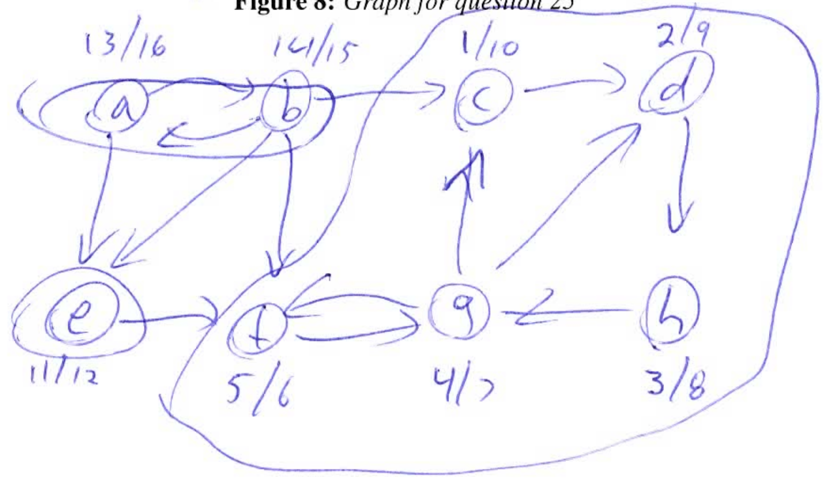


**Figure 8:** *Graph for question 25*



26. Using either Kruskal's or Prim's algorithm, find the minimum spanning tree in Figure 9. List the edges in the minimum spanning tree in the order they are added (5 pts). (Note: this ordering will also be used to determine if you were correctly following the algorithm.)

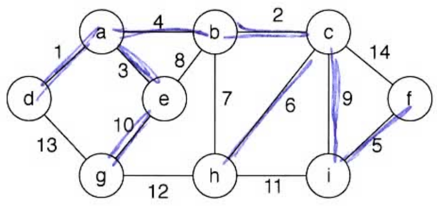


**Figure 9:** *Graph for question 26*

_____ _____ _____ _____ _____ _____ _____ _____ _____

(work space)

prim's
(a, d), (a, e), (a, b), (b, c), (c, h), (c, i), (i, f),
(g, e)

kruskal's
(a, d), (b, c), (a, e), (a, b), (i, f), (c, h), (c, i)
(g, e)

Did you use Kruskal's algorithm or Prim's (1 pt)?