Design and Analysis of Algorithms Final, Fall 2017

Professor Callahan

Name: ____________________________________

NetID: ____________________________________

0A. Did you ever use the code windows provided on the web site?
a. Yes
b. No

0B. Did you ever use the quizzes provided on the web site?
a. Yes
b. No

1. Why is the maximum length of the longest path in a red-black tree double that of the shortest path?
*a. Because the black-height is the same for all leaves, and at most every other node is red.
b. Because the black-height of any leaf is at most twice that of any other.
c. Because the root of the tree is always black as are the leaves.
d. Because insertion takes lg *n* time.

2. Which of the following correctly codes OS-Select(x, i), to get the *i*th ranked node from a tree?
a.
```
r = x.left.size + 1
if i == r
    return x
elseif i < r
    return OS-Select(x.left, i - r)
else return OS-Select(x.right, i - r)
```
b.
```
r = x.right.size + 1
if i == r
    return x
elseif i < r
    return OS-Select(x.left, i)
else return OS-Select(x.right, i - r)
```
*c.

```
r = x.left.size + 1
if i == r
    return x
elseif i < r
    return OS-Select(x.left, i)
else return OS-Select(x.right, i - r)
```
d.
```
r = x.left.size + 1
if i == r
    return x
elseif i < r
    return OS-Select(x.right, i)
else return OS-Select(x.left, i - r)
```

3. For an unweighted graph, a DFS traversal of the graph produces:
*a. a spanning tree
b. an Euler cycle
c. a Hamiltonian cycle
d. both a and c

4. Consider the following adjacency matrix:

	a	b	c	d	e	f	g
a	0	0	1	0	1	0	1
b	0	0	0	0	0	1	0
c	1	0	0	0	0	1	1
d	0	0	0	1	1	0	0
e	1	0	0	1	0	0	0
f	0	1	1	0	0	0	1
g	1	0	1	0	0	1	0

In the graph so described, there is a loop at vertex ___?
a. a
b. b
*c. d
d. f

5. Which of the following is pseudo-code for Interval-Search(), which finds a node in tree ‘T’ whose interval overlaps interval ‘i’.
a.
```
n = T.root
while n != T.nil and i does not overlap n.int
     if n.right != T.nil and n.left.max == i.high
          n = n.right
     else n = n.left
```

```
return n
```
*b.
```
n=T.root
while n != T.nil and i does not overlap n.int
     if n.left != T.nil and n.left.max >= i.low
          n = n.left
     else n = n.right
return n
```
c.
```
n = T.root
while n != T.nil and i overlaps n.int
     if n.right != T.nil and n.left.max >= i.high
          n = n.left
     else n = n.right
return n
```
d.
```
n = T.root
while n == T.nil and i overlaps n.int
     if n.left != T.nil and n.left.max >= i.low
          n = n.left
     else n = n.right
return n.right
```

6. According to the master method, the recurrence $T(n) = 8T(n / 2) + n \lg n$ is
*a. $O(n^3)$
b. $O(n^2)$
c. $O(n \lg n)$
d. The master method is not applicable.

7. If we pass Build-Heap the array [84, 12, 16, 96, 2, 8, 18, 24], the resulting heap will be:
a. [84, 12, 18, 24, 2, 8, 16, 96]
*b. [96, 84, 18, 24, 2, 8, 16, 12]
c. [18, 24, 2, 8, 16, 12, 96, 84]
d. [2, 8, 12, 16, 18, 24, 84, 96]

8. Consider the graph with following adjacency matrix:

	a	b	c	d	e	f
a	0	1	1	1	0	1
b	1	0	1	0	0	0
c	1	1	0	0	0	0
d	1	0	0	0	1	0
e	0	0	0	1	0	1
f	1	0	0	0	1	0

Which of the following is an Eulerian circuit through that graph?
a. a-c-b-a-f-a-d-e-f-a
*b. f-a-b-c-a-d-e-f
c. a-f-e-d-c-b-a
d. a-b-c-d-e-f-a

9. Using just the master theorem, the recurrence $T(n) = 2T(n / 3) + 3T(n / 4)$ is:
*a. not solvable
b. $O(n^2)$
c. $O(n \lg n)$
d. $O(n^4)$

10. Consider the *directed* graph described by the adjacency list {a -> (b, c, d), b -> (e), c -> (b), d -> (f) e: (c), f -> (g), g -> (d)}. What are the *strongly* connected components of that graph?
a. a-b-c-e and d-f-g
*b. a-b-c, e, and d-f-g
*c. a, b-c-e and d-f-g
d. the whole graph is strongly connected

11. How many strongly connected components does the directed graph specified by the following adjacency list have?
Adjacency list: {a -> (b), b -> (d), c -> (b), d -> (c), e -> (a, f, h), f -> (d, g), g -> (e), h -> (g)}
a. 0
b. 1
c. 2
*d. 3

12. If we have a binary tree with 7 levels (including the root), what is the least number of leaves we can have in level 7?
*a. 1
b. 32
c. 64
d. 128

13. Which of the following is a valid prefix coding scheme?
a. a = 01, b = 00, c = 001, d = 011, e = 111
b. a = 001, b = 000, c = 110, d = 01, e = 11
c. a = 001, b = 000, c = 101, d = 00, e = 11
*d. a = 001, b = 000, c = 101, d = 01, e = 11

14. Consider the following Huffman code: a = 111, b = 110, d = 010, f = 10, g = 00, r = 011. What alphabetical string do the digits 110111011101101110 0 represent?

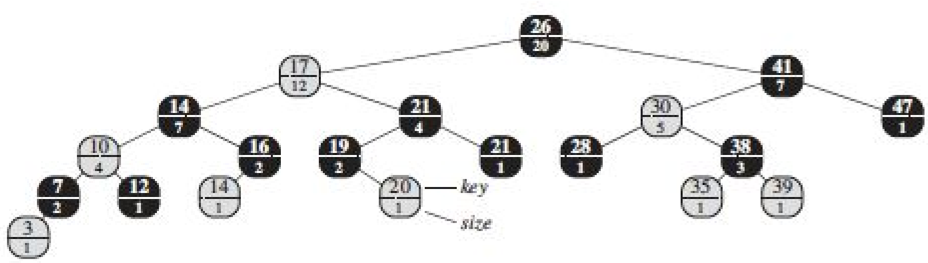
a. barddab
*b. barfbag
c. arfbrag
d. badgarf

15. You are looking at a series of investments that can be made each January. However, financial regulations require you to hold an investment for four years once you are in it, during which time all of your money will be locked up in that investment. Given you have a list, r, of the revenue you expect from each investment, what is the recurrence relation you should use to pick out the maximum revenue you can expect from the entire series?
a. $T(n) = r[n] + T(n - 4)$
b. $T(n) = \max(r[n], T(n - 4))$
*c. $T(n) = \max(r[n] + T(n - 4), T(n - 1))$
d. $T(n) = \max(r[n] + T(n - 1), T(n - 4))$

16. An indicator random variable is actually a:
*a. function
b. constant
c. expectation
d. termination

17. The reason to calculate a node's rank rather than storing it is:
a. it can't be done any other way.
*b. otherwise, inserting a new minimal node would require updating every single node's rank.
c. we can achieve a O(n) runtime that way.
d. none of the above.

18. How does OS-Select(T.root,12) operate on the following red-black tree?



a. Node 26: i=12, rank=13, go to left subtree
   Node 17: i=12, rank=8, go to left subtree
   Node 14: i=4, rank=5, go to right subtree
   Node 16: i=2, rank=2, go to left subtree
   Node 14: i=1, rank=1, return 14
*b. Node 26: i=12, rank =13, go to left subtree
   Node 17: i=12, rank=8, go to right subtree

Node 21: i=4, rank=3, go to right subtree
Node 21: i=1, rank=1, return 21

c. Node 26: i=12, rank =13, go to left subtree
Node 17: i=12, rank=8, go to left subtree
Node 14: i=2, rank=5, go to left subtree
Node 10: i=2, rank=3, go to right subtree
Node 12: i=1, rank=1, return 12

d. Node 26: i=12, rank =13, go to left subtree
Node 17: i=12, rank=8, go to left subtree
Node 14: i=4, rank=5, go to left subtree
Node 10: i=4, rank=3, go to left subtree
Node 7: i=1, rank=1, return 7

19. Consider the following matrix for a weighted graph; in what order will Kruskal's add edges to a minimum spanning tree?

	a	b	c	d	e	f	g	h
a	0	6	0	0	0	0	0	0
b	6	0	2	12	0	0	0	0
c	0	2	0	8	0	14	0	0
d	0	12	8	0	6	3	8	0
e	0	0	0	6	0	0	0	0
f	0	0	14	3	0	0	0	1
g	0	0	0	8	0	0	0	3
h	0	0	0	0	0	1	3	0

a. f-h, b-c, g-h, d-f, a-b, d-e, c-d
b. b-c, b-d, c-b, d-e, f-h, a-b, g-h
c. f-h, b-c, d-f, g-h, d-e, a-b, c-d
*d. a or c

20. Consider the above matrix for a weighted graph: in what order will Prim's algorithm add edges to a minimum spanning tree, if it starts at vertex *a*?
a. a-b, b-c, b-c, d-e, f-h, c-f, g-h
b. a-b, b-d, b-c, c-f, d-e, f-h, h-g
*d. a-b, b-c, c-d, d-f, f-h, g-h, d-e
c. a-b, b-c, c-f, d-e, b-d, f-h, h-g

21. Which of these is a non-recursive version of OS-Select(x, i) (to get the *i*th item in the tree rooted at *x*)?
a.

```
OS-Select(x, i)
    while i != r
      r = x.right.size + 1
      if i < r
          x = x.right
      else
          i = i - r
          x = x.left
    return x
```
b.
```
OS-Select(x, i)
    while i < r
        r = x.left.size + 1
        if i < r
           x = x.left
       else
           i = i - r
           x = x.right
    return x
```
c.
```
OS-Select(x, i)
    while i == r
      r = x.left.size + 1
      if i < r
          x = x.left
      else
          i = i - r
          x = x.right
    return x
```
*d.
```
OS-Select(x, i)
    while i != r
       r = x.left.size + 1
      if i < r
          x = x.left
      else
          i = i - r
          x = x.right
    return x
```

22. Using the master method, the closed form of the recurrence $T(n) = 8T(n / 8) + n \lg n$ is?
a. O(n)
b. $O(n^2)$

c. $O(n \lg n)$
*d. not solvable

23. An ATM can only dispense $1 bills, $5 bills, and $10 bills. How many ways are there to withdraw n dollars (given that we have described our bases suitably)?
a. $F(n) = nF(n) + (n/5)F(n) + (n/10)F(n)$
b. $F(n) = nF(n - 1) + (n/5)F(n - 5) + (n/10)F(n - 10)$
c. $F(n) = F(n - 1) + F(n - 5)^5 + F(n - 10)^{10}$
*d. $F(n) = F(n - 1) + F(n - 5) + F(n - 10)$

24. What is the runtime complexity of the following equation? $T(n) = 5T(n/2) + n^2$
a. $O(n \lg n)$
b. $O(n^2)$
c. $O(n^{\lg n})$
*d. $O(n^{2.32})$

25. If we are using open addressing, and the quadratic hashing function $(((\text{floor}(x/2) + 3i + 2i^2)$ mod $m)$ and a table size (m) of 11, in which slot will the value 19 wind up if we must probe three times, only finding an open slot on the third probe?
a. 2
b. 7
c. 6
*d. 1

26. A thief enters a store and finds the following items: 3 pounds of item A worth $50 per pound, 2 pounds of item B worth $30 per pound, and 1 pound of item C worth $40 per pound. His knapsack holds 5 pounds. What is his maximum profit if he can take any amount of any good he wants?
*a. $220
b. $250
c. $210
d. $150

27. Consider the graph described by the adjacency list {a: (b, c, d, e), b: (a, c), c: (a, b, d, e), d: (a, c)}. Does it contain an Eulerian circuit?
*a. Yes
b. No
c. Not enough information

28. Let $f(x) = f(x - 1) + 3$, for all integers greater than 3, and let $f(3) = 2$. What is $f(8)$?
a. 14
*b. 17

c. 16
d. 20

29. Use the master theorem to give the runtime for the recurrence $T(n) = 2T(n) + 6n - 1$
a. $O(6n)$
b. $O(n)$
c. $O(n \log n)$
*d. The master theorem does not apply.

30. Find the solution for $f(5)$ for the recurrence relation
$f(n) = 2f(n - 1) + 3(2^{n-1})$, where $f(1) = 3$.
a. 144
b. 288
*c. 240
d. 96

31. Consider the following matrix for a weighted graph: at what point will Dijkstra's shortest-path algorithm, looking for the shortest path from a to h, overwrite the previously recorded shortest path with an intermediate node?

	a	b	c	d	e	f	g	h
a	0	8	0	0	0	0	0	0
b	8	0	4	2	0	0	0	0
c	0	4	0	11	0	9	0	0
d	0	2	11	0	5	6	10	0
e	0	0	0	5	0	0	1	0
f	0	0	9	6	0	0	0	7
g	0	0	0	10	1	0	0	9
h	0	0	0	0	0	7	9	0

a. when it finds a new path to b
b. when it finds a new path to e
*c. when it finds a new path to g
d. when it finds a new path to d
e. none of the above.

32. What is the Big-O runtime complexity of the recurrence $T(n) = 9T(n/3) + 12n^4$?
*a. $O(n^4)$
b. $O(n^3)$
c. $O(n!)$
d. $O(n^2)$

33. If a recursive algorithm stores previously calculated values, that is called ____________
a. saving the value property

*b. memoization
c. hashing
d. mapping

34. A greedy algorithm can be used to solve dynamic programming problems when:
a. There are overlapping subproblems.
*b. The cost of a choice is locally contained.
c. The cost of a choice affects other parts of the problem.
d. All of the above.
e. None of the above.

35. If *A* and *B* are both spanning trees for graph *G*, then we know that:
a. both *A* and *B* are Eulerian paths through *G*.
b. neither *A* nor *B* contain a cycle.
c. both *A* and *B* have the same number of edges.
*d. b and c are both true.
e. none of the above are true.

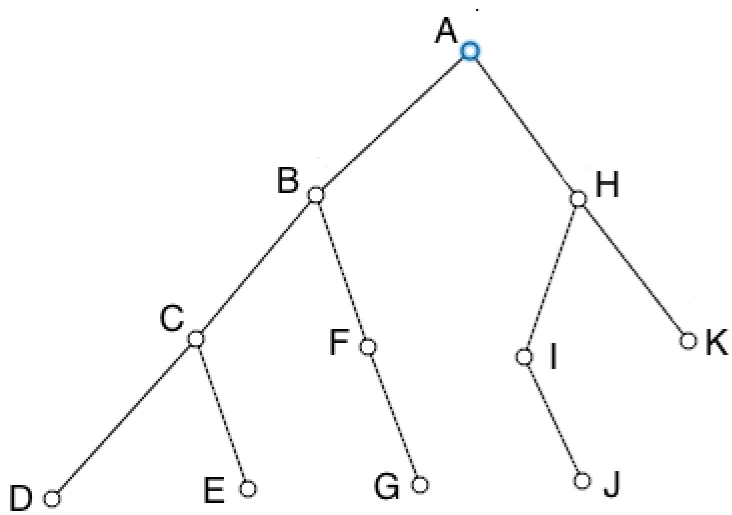
36. Consider a rod of length 7 and the prices for each length:

Length	Price
1	3
2	5
3	10
4	11
5	14
6	17
7	20

What combination of its pieces would give you the maximum price?
a) {1, 4}
*b) {1, 3, 3}
c) {2, 3}
d) {5}

37. When we have an optimal substructure and only the locally optimal choice matters, we should probably use:
a. a straightforward recursive solution.
b. dynamic programming.
c. a memoized recursive algorithm.
*d. a greedy algorithm.

38. A post-order walk of the following tree will print the nodes in what order?



a. D-E-C-F-G-B-J-I-K-H-A
b. A-B-H-C-F-I-K-D-E-G-J
*c. D-E-C-G-F-B-J-I-K-H-A
d. A-B-C-H-F-I-K-D-E-G-J

39. Which of the following pieces of pseudo-code correctly searches a binary tree?
a.
```
Tree-Search(x, k)
if k = x.key
    return x
if k < x.key
    return Tree-Search(x.left, k)
else return Tree-Search(x.right, k)
```
*b.
```
Tree-Search(x, k)
if x == NIL or k = x.key
    return x
if k < x.key
    return Tree-Search(x.left, k)
else return Tree-Search(x.right, k)
```
c.
```
Tree-Search(x, k)
if x == NIL or k = x.key
    return x
```

```
if k < x.key
    return Tree-Search(x.right, k)
else return Tree-Search(x.left, k)
```
d.
```
Tree-Search(x, k)
if x == NIL
    return x
if k < x.key
    return Tree-Search(x.left, k)
else return Tree-Search(x.right, k)
```

40. lg* of 1,048,576 is
a. 20
b. 4
c. 22
*d. 5

41. "Linearity of expectations" allows us to assert which of the following equalities?
*a.
$$E\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} E\left[X_i\right]$$
b.
$$E\left[X\right] = E\left[\sum_{i=1}^{n} X_i\right]$$
c.
$$E\left[\sum_{i=1}^{n} X_i\right] = \frac{1}{i}$$
d.
$$E\left[\sum_{i=1}^{n} X_i\right] = nE\left[\sum_{i=1}^{n} X_i\right]$$

42. Employing the master theorem, the solution to the recurrence $T(n) = 9T(n/3) + n$ is
*a. $\Theta(n^2)$
b. $\Theta(n^2 \lg n)$
c. $\Theta(n)$
d. the master theorem can't be applied here.

43. You are taking a long car trip, and you know that your child needs to stop and use the bathroom every *m* miles. You need to plan your rest stops carefully to you stop the fewest possible times, but so that there are no "accidents" in the back seat. If S is going to store the set of stops you will make, s[i] is the array of all possible stops, and d[i] is the distance from the start to s[i], which code will correctly pick the stops you need to make?

a.
```
S = NIL
last = 0
for i = 1 to n
    if (d[i] + last) > m
        S = S Union {s[i - 1]}
        last = d[i - 1]
```
b.
```
S = NIL
last = 0
for i = 1 to n
    if (d[i] - last) > m
        S = S Union {s[i]}
        last = d[i] - 1
```
c.
```
S = NIL
last = 0
for i = 1 to n
    if (d[i] - last) > m
        S = S Union {s[i] - 1}
        last = d[i]
```
*d.
```
S = NIL
last = 0
for i = 1 to n
    if (d[i] - last) > m
        S = S Union {s[i - 1]}
        last = d[i - 1]
```

44. Using the master theorem, the solution to the recurrence
T(n) = $64T(n/8) - n^2$ is:
a. $\Theta(n^2)$
b. $\Theta(n^4 \lg n)$
*c. $\Theta(n^2 \lg n)$
d. the master theorem can't be applied here.

45. Which of the following pieces of code correctly implements successor for a binary search tree?

*a.

```
if x.right != NIL
     return Tree-Minimum(x.right)
y = x.p
while y != NIL and x == y.right
     x = y
     y = y.p
return y
```

b.

```
if x.left != NIL
     return Tree-Minimum(x.left)
y = x.p
while y != NIL and x == y.right
     x = y
     y = y.p
return y
```

c.

```
if x.right != NIL
     return Tree-Minimum(x.right)
y = x.p
while y != NIL and x == y.left
     x = y
     y = y.p
return y
```

*d.

```
if x.right != NIL
     return Tree-Minimum(x.right)
y = x.p
while y != NIL and x == y.right
     x = y
     y = y.p
return y
```

46. If we have a hash table with chaining of size m = 17 and our hash function is $h(x) = x \bmod m$, if we hash the numbers 12, 45, 13, 16, 78, 44, 2, 3, 4, 23, 32, 22, 35, 1, and 52 how many items will be in the linked list at slot 10?

*a. 2

b. 1

c. 0

d. 4

47. If we were using open addressing and linear probing, still with $m = 17$ and input as in the previous question, in what slot will the number 44 wind up?
a. 44
b. 10
*c. 14
d. 0

48. Disjoint sets, or union-find, would be useful in:
a. a binary search tree
*b. Kruskal's algorithm
c. quicksort
d. heapsort

49. The disjoint set ADT consists of the operations
a. union, intersection, complement
*b. make-set, find, union
c. make-set, disjoint-set, add-set
d. make-set, branch-set, fork-set

50. If we randomize an algorithm, then we can speak of its
a. excessive running time
b. average running time
*c. expected running time
d. none of the above

51. Given a hash table with 44 slots, which stores 2800 elements, the load factor α is:
a. .0157
b. .30
c. 116.66
*d. 63.63

52. Consider a hash table of size nine, with starting index zero, and a hash function $(2x + 4) \bmod 9$. Assuming the hash table is initially empty, which of the following is the contents of the table when the sequence 12, 4, 7, 16, and 25 is inserted into the table using open addressing with linear probing? Note: '_' denotes an empty location in the table.
a. 7, 25, 4, 16, 12, _, _, _, _
*b. 7, 12, 16, 4, 25, _, _, _, _
c. 12, 7, _, 4, 16, 25, _, _, _
d. 16, 7, 12, _, _4, _, 25, _, _

53. What red-black tree operation does the following C code implement?

```
void f(struct node *n)
{
    struct node *nnew = n->right;
    assert(nnew != LEAF);
    n->right = nnew->left;
    nnew->left = n;
    nnew->parent = n->parent;
    n->parent = nnew;
    // (the other related parent and child links would also have to
be updated)
}
```

a. rb-insert()
b. rb-delete()
c. rotate-right()
*d. rotate-left()

54. Greedy algorithms are also used widely in memory management. Given a list of blocks, in which the sizes are { 200, 40, 300, 60, 240 }, how would a greedy algorithm (grab the smallest block that works first) fit processes with sizes {10, 260, 180, 80, 170 }?
a. process 1: block 1, process 2: block 2, process 3: block 4, process 4: block 3, process 5: block 5
*b. process 1: block 2, process 2: block 3, process 3: block 1, process 4: block 5, process 5: can't run
c. process 1: block 5, process 2: block 2, process 3: block 4, process 4: block 3, process 5: block 1
d. process 1: block 2, process 2: block 5, process 3: block 4, process 4: block 2, process 5: can't run

55. "Optimal substructure" in dynamic programming means
*a. the optimal solution to each subproblem can be counted on to be part of the overall optimal solution.
b. each subproblem must be structured optimally before the algorithm can run.
c. the code for the algorithm must have each line elegantly crafted for optimal performance
d. all of the above

56. Huffman’s algorithm is a greedy algorithm because:
a. The frequencies are combined at each step
b. It involves drawing a Huffman tree
*c. The two smallest probability nodes are chosen at each step
d. Both a and c.

57. Consider a max heap, represented by the array: [96, 84, 18, 24, 2, 8, 16, 12] The value 20 is inserted into this heap. After insertion and heapification, the new heap is:
a. [96, 84, 18, 24, 20, 2, 8, 16, 12]
*b. [96, 84, 18, 24, 2, 8, 16, 12, 20]
c. [96, 84, 18, 24, 2, 8, 20, 16, 12]
d. [96, 84, 20, 18, 24, 2, 8, 16, 12]

58. When multiplying an $m * n$ matrix with an $n * p$ matrix, the straightforward run-time complexity (without using something like Strassen's algorithm) is:
*a. $m * n * p$
b. $m * p$
c. $n * p$
d. $m * \lg n * p$

59. A common alternative to bottom-up dynamic programming, which performs roughly as well, is:
*a. a memoized recursive algorithm
b. a binary search tree
c. a greedy algorithm
d. an exponential algorithm

60. What are the worst case and average case complexities of searching in a binary search tree?
a. $O(n^2)$, $O(n)$
b. $O(n \lg n)$, $O(n)$
c. $O(\lg n^2)$, $O(\lg n)$
*d. $O(n)$, $O(\lg n)$

61. The worst case scenario for binary search trees occurs when
a. the input is sorted
b. the input is completely random
c. the input is sorted in reverse order
*d. both a and c

62. What does the following pseudo-code do?
```
f(T)
    x = T.root
    while x.right is not NIL
        x = x.right
    return x
```
a. find the minimum item in a BST
*b. find the maximum item in a BST
c. find the successor of T in a BST

d. find the predecessor of T in a BST

63. The *purpose* of the red-black tree rules is
a. to ensure the root of the tree is always black
b. to keep all leaves black
c. to ensure red nodes have no red children
*d. to keep a BST balanced through local operations

64. In an augmented red-black tree keeping *i*th-order statistics, the rank of a node is
a. the size of its right subtree plus one
b. the size of its left subtree plus one
*c. the size of its left subtree plus one, plus the size of any left subtrees of nodes of which it is in the right subtree on the way to the root
d. the size of its left subtree plus one, plus the size of any right subtrees of nodes of which it is in the left subtree on the way to the root

65. A greedy algorithm can be used to solve dynamic programming problems when:
a. There are overlapping subproblems.
b. The cost of a choice affects other parts of the problem.
*c. The cost of a choice is locally contained.
d. All of the above.

66. The recurrence $T(n) = 2T(n/2) + n$ has the runtime complexity of $n \lg n$ because:
*a. n work is done at each of $\lg n$ levels.
b. $\lg n$ work is done at each of n levels.
c. $n \lg n$ work is done at each level.
d. $n \lg n$ is the number of levels of the recurrence.

67. An example of an abstract data type (ADT) is:
a. a string
b. a floating-point number
c. a 64-bit integer
*d. a dictionary

68. We have one algorithm for processing customer records with runtime of $O(n)$, and another with runtime of $O(\lg n) + 100$. In what circumstances might we want to choose the $O(n)$ algorithm?
a. No circumstances.
*b. We believe our program will always be dealing with a number of records less than 100.
c. If our programmers are really bad.
d. If n is very large.

69. If we want to show that $5n^3 + 10n + 17$ is $\Theta(n^3)$, we could choose constant k_1 (for upper bound), k_2 (for lower bound) and n_0 as:
a. $k_1 = 2$, $k_2 = .5$, $n_0 = 1000$
b. $k_1 = 4$, $k_2 = 2$, $n_0 = 5000$
c. $k_1 = 6$, $k_2 = 4$, $n_0 = 100$
*d. $k_1 = 6$, $k_2 = 4$, $n_0 = 6000$

70. If we set up an indicator random variable *X* for "a fair, six-sided die comes up 6" to equal 1 if the result occurs, and 0 if it does not, then the expected value of the sum of *X* over 60 tosses is:
a. 1/6
*b. 10
c. 6
d. 20

71. Which piece of code below correctly implements Tree-Delete() for a binary search tree, assuming we have a working version of Transplant(T, u, v) which puts *v* in place of *u* in *T*?
a.
```
Tree-Delete(T, z)
    if z.left == NIL
        Transplant(T, z, z.left)
    elseif z.right == NIL
        Transplant(T, z, z.right)
    else
        y = Tree-Minimum(z.right)
        if y.p != z
            Transplant(T, y, y.right)
             y.right = z.right
             y.right.p = y
        Transplant(T, z, y)
        y.left = z.left
        y.left.p = y
```
b.
```
Tree-Delete(T, z)
    if z.left == NIL
        Transplant(T, z, z.right)
    elseif z.right == NIL
        Transplant(T, z, z.left)
    else
        y = Tree-Maximum(z.right)
        if y.p != z
            Transplant(T, y, y.right)
             y.right = z.right
             y.right.p = y
```

```
        Transplant(T, z, y)
        y.left = z.left
        y.left.p = y
```
*c.
```
Tree-Delete(T, z)
    if z.left == NIL
        Transplant(T, z, z.right)
    elseif z.right == NIL
        Transplant(T, z, z.left)
    else
        y = Tree-Minimum(z.right)
        if y.p != z
            Transplant(T, y, y.right)
             y.right = z.right
             y.right.p = y
        Transplant(T, z, y)
        y.left = z.left
        y.left.p = y
```
d. none of the above

72. Employing the master theorem, the solution to the recurrence $T(n) = 2T(n/4) + n^{0.49}$ is
*a. $\Theta(n^{1/2})$
b. $\Theta(n^2 \lg n)$
c. $\Theta(n^{0.49})$
d. the master theorem can't be applied here.

73. The worst-case running time of a search on a binary search tree is
a. $O(n^2)$
*b. $O(n)$
c. $O(\lg n)$
d. $O(n \lg n)$

74. For depth-first search, the discovery times for nodes and their finish times have a
a. comma structure
*b. parenthesis structure
c. ellipsis structure
d. semi-colon structure