

I have pulled questions from a variety of exams to fit our current coverage. So your exam may differ from this one in length.

## Multiple Choice (3 points each)

1. For the timing information below, what is the complexity?

n	T(n)
2	1
4	2
8	4
16	9
32	17

- (a)  $O(1)$  (b)  $O(\log n)$  (c)  $O(n)$  (d)  $O(n \log n)$  (e)  $O(n^2)$
2. For the timing information below, what is the complexity?

n	T(n)
2	1
4	2
8	4
16	4
32	4

- (a)  $O(1)$  (b)  $O(\log n)$  (c)  $O(n)$  (d)  $O(n \log n)$  (e)  $O(n^2)$
3. Algorithm C has complexity  $O(n)$ . What do you expect to happen to the execution time if the problem size doubles?
1. doubles
  2. slightly more than doubles
  3. quadruples
  4. increases by a constant
4. Algorithm D has complexity  $O(n^2)$ . What do you expect to happen to the execution time if the problem size doubles?
1. stays the same
  2. doubles
  3. slightly more than doubles
  4. quadruples
  5. increases by a constant

5. You have two solutions for the same problem. One is  $O(n^2)$  and the other is  $O(n \log n)$ . When you compare the times for various problems, the  $O(n \log n)$  algorithm is always significantly *slower*. How do you explain this?
1. The clock is not accurate.
  2. Your problem sizes are too small to see the advantages of the  $O(n \log n)$  algorithm.
  3. You are running on a very fast machine.
  4. The  $O(n^2)$  algorithm has a higher overhead.

6. From our theorem we know:

$$T(n) = a T(n/b) + O(n^k)$$

$$a > b^k$$

$$\text{if } T(n) \text{ is } O(n^{\log_b a})$$

$$\text{if } a = b^k T(n) \text{ is } O(n^k \log n)$$

$$a < b^k$$

$$\text{if } T(n) \text{ is } O(n^k)$$

Consider the following algorithm:

```
int doit(int A[], int n){
    if (n <=1) return 1;
    int t;
    for (int i =0; i < n; i++)
        t++;
    t+= 3*doit(A,n/2)
    for (int i =0; i < n; i++)
        t++;
    return t;
}
```

What is the complexity?

- (a)  $O(1)$  (b)  $O(\log n)$  (c)  $O(n)$  (d)  $O(n \log n)$  (e)  $O(n^2)$

7. For the problem above, what are the values of a, b and k?
1. a=1, b=1, k=1
  2. a=1, b=2, k=1
  3. a=1, b=2, k=0
  4. a=3, b=2, k=2
8. A breadth first traversal of a graph is commonly done using
- a. a stack
  - b. a queue
  - c. a topological ordering
  - d. a priority queue
  - e. none of the above
9. A depth first traversal, visiting all n nodes of a tree has what complexity?
- a.  $O(n \log n)$
  - b.  $O(n)$
  - c.  $O(\log n)$

- d.  $O(1)$
- e. none of the above

10. If  $G$  is a directed graph with 20 vertices, how many boolean values will be needed to represent  $G$  using an adjacency matrix?

- a. 20
- b. 40
- c. 200
- d. 400

11. How many linked lists are used to represent a graph with  $n$  nodes and  $m$  edges, when using an edge list representation?

- a.  $m$
- b.  $n$
- c.  $m + n$
- d.  $m * n$

12. In a selection sort of  $n$  elements, how many times is the swap function called in the complete execution of the algorithm?

- a. 1
- b.  $n - 1$
- c.  $n \log n$
- d.  $n^2$

13. When is insertion sort a good choice for sorting an array?

- a. Each component of the array requires a large amount of memory.
- b. Each component of the array requires a small amount of memory.
- c. The array has only a few items out of place.
- d. The processor speed is fast.

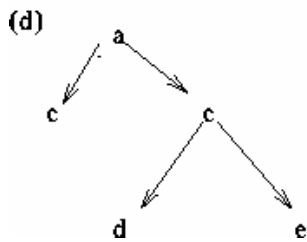
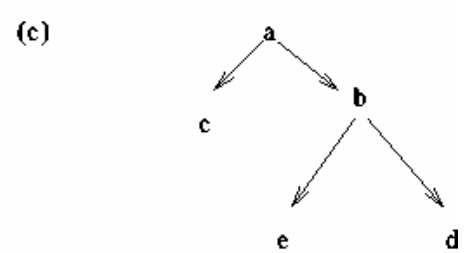
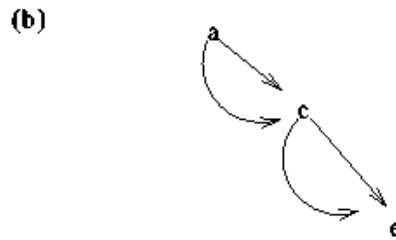
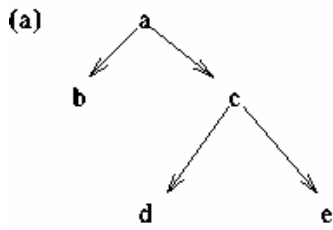
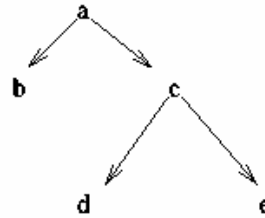
14. Consider the following code on a binary tree. What is the result of the call `doit(Tree1)`?

```

void doit(Node * t)
( if (t==NULL) return;
  t->left= t->right;
  doit(t->left);
  doit(t->right);
)

```

Tree1



(e) None of the above

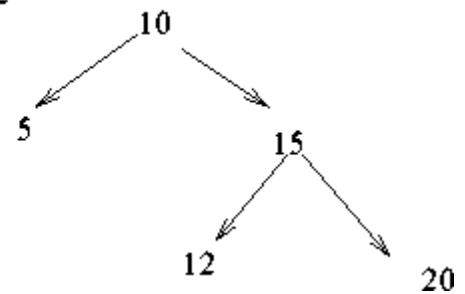
15. For the code below, what is the result of a call to also(Tree2)?

```

int also(Node *t)
( if (t==NULL) return 0;
  return t->val + also(t->left)+ also(t->right);
)

```

Tree2



(a) 10 (b) 12 (c) 62 (d) 37 (e) none of the above

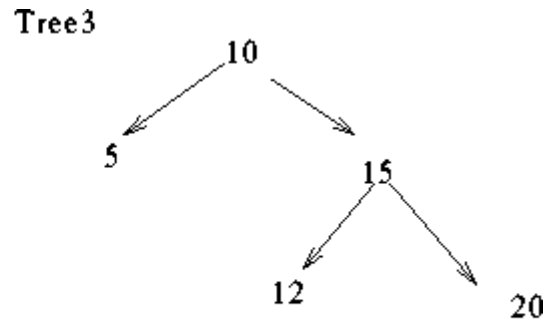
16. For the code below, what is the result of a call to compute(Tree3)?

```

int compute(Node * t)
{ if (t==NULL) return 0;
  if (t->left ==NULL && t->right ==NULL)
    return 1;

  return compute(t->left) + compute (t->right);
}

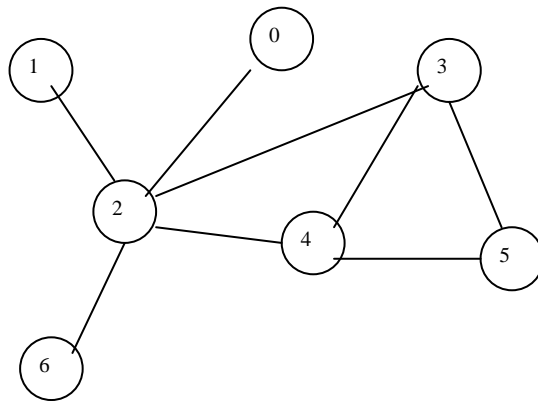
```



- (a) 3 (b) 5 (c) 0 (d) 1 (e) none of the above

### Short Answer

1. (5 points) Consider this graph. In what order could the vertices be visited for a depth-first search that starts at 0?



1.  
2. (15 points ) Find the complexity of the following pieces of code. For each problem, draw an appropriate picture to justify your answer.

```

(a) void doit(int n)
{ if (n/2 <=1) return;
  int x = 0;
  for (int i = 0; i < n; i++) x++;
  doit(n/2);
}

```

```

(b) void doit(int n)
{ if (n/2 <=1) return;
  doit(n/2);
  doit(n/2);
}

```

```

(c) void doit(int n)
{ if (n <=0) return;
  for (int i=0;i<n;i++)
    for (int j=0;j<i;j++)

```

```

        cout << i*j;
    }

```

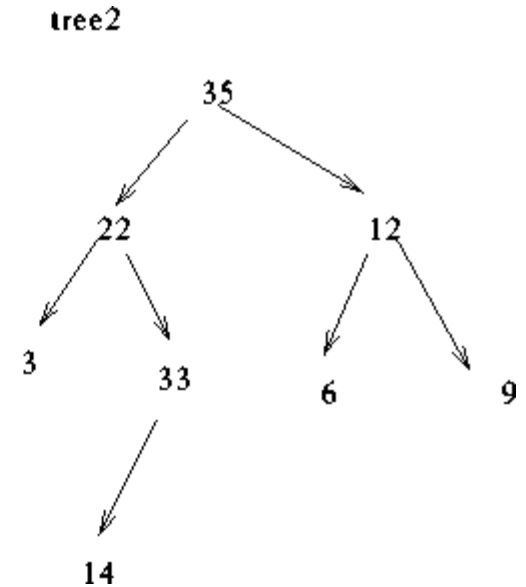
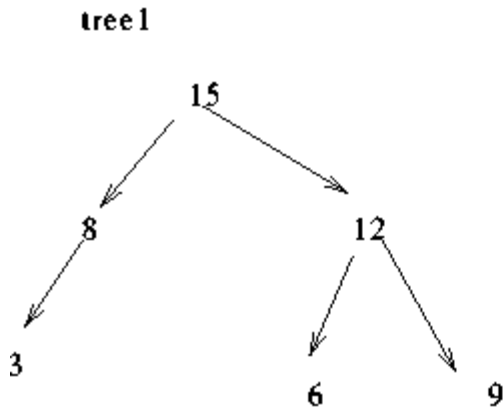
3. (20 points) Given a binary tree, write the function which returns TRUE if every node in a tree is bigger in value than it's children.

For example, for the following trees

```

isBig(tree1) = true;
isBig(tree2) = false;

```



4. (14 points) Answer the following questions about the code below.

```

void swap(int a[], int i, int j)
{
    int temp= a[i];

    a[i] = a[j];

    a[j]=a[i];
}

```

```

void sort(int a[], int size)
{
    int big = size;
    for (int i=0; i <size; i++)
        if (a[big] < a[i]) big = i;
    swap(a,size,big);
    sort(a,size-1);
}

```

1. What type of sort is coded?
2. Correct any errors.
3. What does size represent?
4. Is the sort stable? Explain.
5. Is the sort oblivious? Explain.