[redacted]

1. Which one of the following gives the tightest bound on the time complexity of the following code:

 int i, j, k = 0,sum=0;
for (i = n / 2; i <= n; i++){
for (j = 1; j <= n; j = j * 2) {
 k = k + n / 2;
 sum = sum + k + i + j;
 }
 }
for (i= n/2; i<= n ; i++) {
 sum = sum - i;
 }
 Correct

- ( ) O(n)
- (•) O(n * logn)
- ( ) O(n^2)
- ( ) O(n^2 * logn)

2. The recurrence equation
T(1) = 1
T(n) = 2T(n - 1) + 1, n ≥ 2
evaluates to **Correct**

- ( ) 2^(n - 1) + 1
- (•) 2^n - 1
- ( ) 2^(n + 1) - 1
- ( ) 2^n - n

3. The running time of an algorithm is represented by the following recurrence relation:

 if n <= 3 then T(n) = n <br/> else T(n) = T(n/3) + cn

Which one of the following gives the tightest bound on the time complexity of the algorithm? **Correct**

- (•) O(n)
- ( ) O(n log n)
- ( ) O(n^2)
- ( ) O(n^2 log n)

4. Let a(n) be the number of n-bit strings that do NOT contain two consecutive 1s. Which one of the following is the recurrence relation for a(n)? **Correct**

- [ ] a(n)=a(n−1)+2a(n−2)
- [x] a(n)=a(n−1)+a(n−2)
- [ ] a(n)=2a(n−1)+a(n−2)
- [ ] a(n)=2a(n−1)+2a(n−2)

5. Let the minimum number of comparisons to find the minimum of 150 numbers be x and the minimum number of comparisons to find the maximum of 150 numbers be y. Then which of the following is greater than x+y. **Correct**

- [ ] 290
- [ ] 295
- [x] 300
- [x] 310
- [ ] None

6. In the following function, let n ≥ m.

int gcd(n,m) {
 if (n%m ==0) return m;
 n = n%m;
 return gcd(m,n);
}

Which is the tightest bound on the time complexity for gcd(n,m)? **Correct**

- (•) O(log(m))
- ( ) O(m)
- ( ) O(log(log(m)))
- ( ) O(log(log(n)))

7. The worst case occurs in following linear search algorithm when
linearSearch(int a[], int toFind, int length){
 flag = false;
 i=0;
while(i < length and flag== false){
 if(a[i]==toFind){
 flag = true;
 }
 i++;
}
return flag;
}
 Correct

- [ ] Item is in the middle of the array
- [x] Item is the last element of the array
- [x] Item is not present in the array
- [ ] Item is the first element of the array

8. What is the worst case time complexity of inserting a node in a doubly linked list? **Correct**

- ( ) O(nlogn)
- ( ) O(logn)
- ( ) O(n)
- (•) O(1)

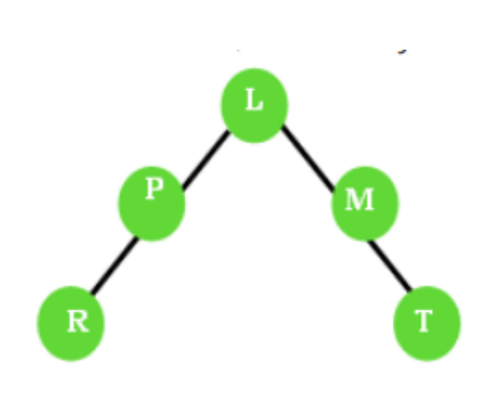
9. Consider the following doubly linked list: (1,2,3,4,5). What will be the 3rd element in the list after performing the given sequence of operations?

 Insert(6,head,head), Insert(0,head.right.right.right, head), Delete (head.right.right, head) **Correct**

- (•) 0
- ( ) 1
- ( ) 2
- ( ) 3
- ( ) 4
- ( ) 5
- ( ) 6

10. Figure below is a perfectly balanced binary tree. If a node is inserted as a right child of the node R, how many nodes will become unbalanced? **Correct**

- ( ) 2
- (•) 1
- ( ) 3
- ( ) 0
- ( ) 4
- ( ) 5



11. Suppose we define an almost balanced binary search tree to have at most 9n/10 many nodes in either the left or right subtree. Then the tightest upper bound on the height of the tree is **Correct**

- ( ) log_(9/10) n
- (•) log _(10/9) n
- ( ) log_2 10n/9
- ( ) log_2 9/10n

12. For an internal node of a BST g(x) is defined as <br/> g(x) = min{no. of leaf-nodes in left-subtree of x, no. of leaf-nodes in right-subtree of x}.
 A program takes as input a binary search tree with n nodes and computes the value g(x) for each internal node x. The tightest time complexity of the program is **Correct**

- ( ) O(logn)
- ( ) O(nLogn)
- ( ) O(n^2)
- (•) O(n)

13. The tightest bound on the worst case running time to search for an element in a perfectly balanced binary search tree with n*2^n elements is **Correct**

- ( ) O(nlogn)
- ( ) O(n * 2^n)
- (•) O(n)
- ( ) O(logn)

14. Solve the recurrence relation T (n) = T (n − 1) + n^c , where, c > 1 is a constant. **Correct**

- (•) T (n) = O(n^(c+1))
- ( ) T (n) = O(n^(c-1))
- ( ) T (n) = O(n^(c))
- ( ) T (n) = O(n^(c)logn)

15. Which of the following algorithms is NOT a divide & conquer algorithm by nature? **Correct**

- [ ] Merge Sort
- [ ] Quick Sort
- [x] Insertion Sort
- [ ] Binary Search
- [x] Selection Sort

16. Consider the polynomial p(x) = a0 + a1x + a2x^2 +a3x^3, where ai != 0, for all i. The minimum number of multiplications needed to evaluate p on an input x is: **Correct**

- ( ) 2
- (•) 3
- ( ) 4
- ( ) 5
- ( ) 6
- ( ) 7
- ( ) 8
- ( ) 9

17. Consider a situation where you don't have a function to calculate power (pow() function in C) and you need to calculate x^n where x can be any number and n is a positive integer. What can be the best possible time complexity of your power function? **Correct**

- ( ) O(n)
- ( ) O(nLogn)
- ( ) O(LogLogn)
- (•) O(Logn)

18. An array(a1 a2.....ak....an) is said to be rotated by k if it becomes of the form (a(k+1)......an a1....ak). Consider the problem of searching an element x in an array of size n. The problem can be solved in O(Logn) time if **Correct**

- [x] the Array is sorted
- [x] the Array is sorted and rotated by k. k is given to you and k <= n
- [x] the Array is sorted and rotated by k. k is NOT given to you and k <= n
- [ ] the Array is not sorted

19. What is the time complexity of the following code:

 int a = 0, b = 0;
 for (i = 0; i < N; i=i+2) {
 a = a + rand();
 }
 for (j = 0; j < M; j=j*2) {
 b = b + rand();
 }
 Correct

- ( ) O(N * logM)
- ( ) O(logN + M)
- (•) O(N + log M)
- ( ) O(logN + logM)

20. What is the time complexity of following code:

 int a = 0;
 for (i = 0; i < N; i++) {
 for (j = N; j > i; j--) {
 a = a + i + j;
 }
}
 Correct

- ( ) O(N)
- ( ) O(N*log(N))
- ( ) O(N * Sqrt(N))
- (•) O(N*N)

21. What does it mean when we say that an algorithm X is asymptotically more efficient than algorithm Y? **Correct**

- [ ] X will always be a better choice for small inputs
- [x] X will always be a better choice for large inputs
- [ ] Y will always be a better choice for small inputs
- [ ] X will always be a better choice for all inputs

22. What is the time complexity of following code:

int a = 0, i = N;
while (i > 0) {
 a =a+ i;
 i =i/4;
}
 Correct

- ( ) O(N^(1/4))
- ( ) O(Sqrt(N))
- ( ) O(N / 4)
- (•) O(log N)
- ( ) O(N)

23. Which one of the following is an application of Stack Data Structure? **Correct**

- [x] Managing function calls
- [ ] Binary search
- [x] Arithmetic expression evaluation
- [ ] Managing a Telephone directory

24. If an arithmetic expression (without parenthesis) has n operands and m operators then the maximum height of the operand stack (N-stack) and operator stack (O-stack) respectively are **Correct**

- ( ) m and n
- ( ) n and m
- (•) m+n and m
- ( ) 2n and m

25. What is the worst case time complexity for search, insert and delete operations in a general Binary Search Tree? **Correct**

- (•) O(n) for all
- ( ) O(Logn) for all
- ( ) O(Logn) for search and insert, and O(n) for delete
- ( ) O(Logn) for search, and O(n) for insert and delete

26. We are given a set of n distinct elements and an unlabeled binary tree with n nodes. In how many ways can we populate the tree with the given set so that it becomes a binary search tree? **Correct**

- ( ) 0
- (•) 1
- ( ) n!
- ( ) (1/(n+1)).2nCn

27. How many distinct binary search trees can be created out of 4 distinct keys? **Correct**

- ( ) 4
- (•) 14
- ( ) 1
- ( ) 42

28. Consider the following two statements about a binary search tree

 I: If u is a child of a parent of v, then u and v must be the same.
 II: If u is a parent of a child of v, then u and v must be the same.

 Then which of the above statements are true? **Correct**

- ( ) Both I and II are true
- ( ) I is true and II is false
- (•) I is false and II is true
- ( ) Both I and II are false

29. Suppose we evaluate the following arithmetic expression using the algorithm discussed in class:
 8/2^3+2*3-5*1

 The element at the top of the operand stack (N-stack) when the '-' operator is popped out is **Correct**

5

30. Following is C like pseudo code of a function that takes a number as an argument, and uses a stack S to do processing.

void fun(int n) {
 Stack S; // Say it creates an empty stack S
while (n > 0) {
 // This line pushes the value of n%2 to stack S <br/> push(S, n%2);
 n = n/2;
 }
 // Run while Stack S is not empty <br/> while (!isEmpty(S))
 printf("%d ", pop(S));
// pop an element from S and print it
}

 What does the above function do in general? **Correct**

- [ ] Prints binary representation of n in reverse order
- [x] Prints binary representation of n
- [ ] Prints the value of Log n
- [ ] Prints the value of Log n in reverse order

31. Consider the following C program

 int main() {
 int x, y, m, n;
 scanf ("%d %d", &x, &y); /* x > 0 and y > 0 */
 m = x; n = y;
 while (m != n) {
 if(m>n) m = m - n;
 else n = n - m;
 }
printf("%d", n);
}

 What does the program compute? **Correct**

- ( ) x + y using repeated subtraction
- ( ) x mod y using repeated subtraction
- (•) the greatest common divisor of x and y
- ( ) the least common multiple of x and y

32. State whether the following statement is true or false. n^2 =O(2^n) **Correct**

- (•) True
- ( ) False

33. State whether the following statement is true or false. n^3 = O(n^4 log n) **Correct**

- (•) True
- ( ) False

34. State whether the following statement is true or false. n^3 = O(n^2) **Correct**

- ( ) True
- (•) False

35. State whether the following statement is true or false. n! = O(3^n) **Correct**

- ( ) True
- (•) False

36. State whether the following statement is true or false. 2^n = O(2^(n+c)), for any constant c>0. **Correct**

- (•) True
- ( ) False

37. Consider the above algorithm for determining whether a sequence of parentheses is balanced.

 balancedParentheses(str){
CreateEmptyStack(stack);
 i=0;
n=length(str);
 while(i < n){
 if(str[i]=='(') Push(stack,'(');
else {
 If (IsEmpty(stack)) return false;
Pop(stack);
}
 i=i+1;
}
return IsEmpty(stack);
}

 What is the maximum number of parentheses that appear on the stack AT ANY TIME when the algorithm analyzes: (()(())(())))? **Correct**

- ( ) 1
- ( ) 2
- (•) 3
- ( ) 4 or More