1. (2 pts) **Memory leak**:
    a. What is memory leak?
    b. Write a C/C++ function that has memory leak problem.

2. (4 pts) **Shallow copy**: A program is used to record each student's quiz scores.

    Answer the following sub-questions:

    a. Class definition:

```
class student {
        public:
                student(int n=5){count=n; score=new int[count];};
                ~student(){delete [] score;};
                int *score, count;
                string name;
};
```

    b. Main program:

```
int main(){
        student a(3); a.name="John"; a.score[0]=70; a.score[1]=60; a.score[2]=30;
        student b=a; b.name="Mary"; b.score[2]=90;
}
```

    a. What are the contents of variables a and b?
    b. What are the two potential problems of this program?

3. (4 pts) **STL vectors**: Give 3 advantages and 1 disadvantage of STL vectors when compared with the standard C/C++ arrays.

4. (3 pts) **STL vectors**: Given an STL vector x, answer the following sub-questions.
    a. What does "x.reserve(25)" mean?
    b. What is the difference between x[i] and x.at(i)?
    c. What is the difference between x.size() and x.capacity()?

1. (2 pts) **Row-based vs. column-based operations**: There are two ways to compute the total of elements in a traditional two-dimensional arrays in C/C++, as shown next.
    a. Compute the column sum first, then compute the overall total.
    b. Compute the row sum first, then compute the overall total.

    In general, which one is faster? Why?

2. (3 pts) **Insertion sort**: Show each major step of insertion sort on the sequence {7, 2, 8, 5, 6, 9, 4, 1, 3}.

3. (2 pts) **Binary search**: There are two ways to compute the middle position during a binary search, as shown next.
    a. mid=(left+right)/2
    b. mid=left+(right-left)/2

    What is the advantage of each method?

4. (3 pts) **Search complexity**: What is the time complexity of linear search, binary search, and hash search,and when are they used?

5. (4 pts) **Multiple-key binary search**: If you want to do binary search over multiple keys, what do you need to do in the following two stages:
    - Preprocessing stage
    - Search stage

6. What else can we use to compute mid?

1. (3 pts) **Definition of big-Oh**: Given functions $f(n)$ and g(n), under what condition do we we say that $f(n)$ is O(g(n)) (Note that $g(n)$ is one of the basic functions, such as n^3.)

2. (3 pts) **Example of big-Oh**: Use the definition of big Oh to explain why $5n^3+2(n+10)^2+(n+50)\log(n+20)$ is O(n^3).

3. (4 pts) **Tries**: Given a key sequence of {stop, steer, skip, step, top, took}, draw the trie data structure. (You need to indicate the end-of-key nodes clearly.)

4. (3 pts) **Binary search vs. tries**: What are the complexities of tries with n keys, each with $m$ characters?

    a. Time complexity for construction
    b. Time complexity for Search
    c. Space complexity

5. (4 pts) **Associative arrays in STL**: There are two implementations of associative arrays (dictionaries) in STL, including <map> and <unordered_map>. Explain at least two differences between these two implementations.

6. Prove that F(n)<2^n where F(n)is the Fibonacci series

1. (2 pts) **Advantages of postfix notation**: What are the two major advantages of using postfix notation?

2. (4 pts) **Infix to postfix**: Use a tabular form to show how to convert the following infix notation into postfix.

   f-((a+b)*c+d*e)

   (Note that you need to show the contents of the stack at each step.)

3. (2 pts) **BFS vs. DFS**: In a constraint satisfaction problem for search in a game (such as 2048 or nonograms), why we do we prefer to use DFS (depth-first search) instead of BFS (breadth-first search)?

4. (2 pts) **BFS vs. DFS**: Is DFS more related to stacks or queues? How about BFS?

5. (4 pts) **Evaluation of postfix notation**: Use a tabular form to evaluate the following postfix notation:

   abc-d+/ea-*c*

   (Note that you need to show the contents of the stack at each step.)

1 (5 pts) **About BT**: Given a binary tree and a proper binary tree with the root located at depth 0, the height of the tree is defined as the maximum depth.

a What is the maximum number of nodes at depth d of a binary tree?

b What is the maximum number of nodes in a binary tree of height *h*?

c If a binary tree has 25 nodes, what is its maximum height?

d If a binary tree has 25 nodes, what is its minimum height?

e If a proper binary tree has 35 nodes, what are the numbers of external nodes?

2 (2 pts) **Vector representation for BT**: In order to access the tree nodes quickly, the following binary tree is stored level-by-levey in a one-dimensional array A. What are the indices for nodes L and M, respectively?

3 (3 pts) **Traversal of BT**: Give the preorder, inorder, and postorder traversals of the following expression tree.

4 (3 pts) **BT from inorder and preorder sequences**: Draw a binary tree with inorder sequence [a b c d e f g h i] and preorder sequence [f b a e d c h g i].

5 (3 pts) **BT from inorder and postorder sequences**: Draw a binary tree with inorder sequence [a b c d e f g h i] and postorder sequence [b a e d g i h f c].

6(3 pts) **Conversion to BT**: Use the left child-right sibling representation to convert the following tree into a binary tree.

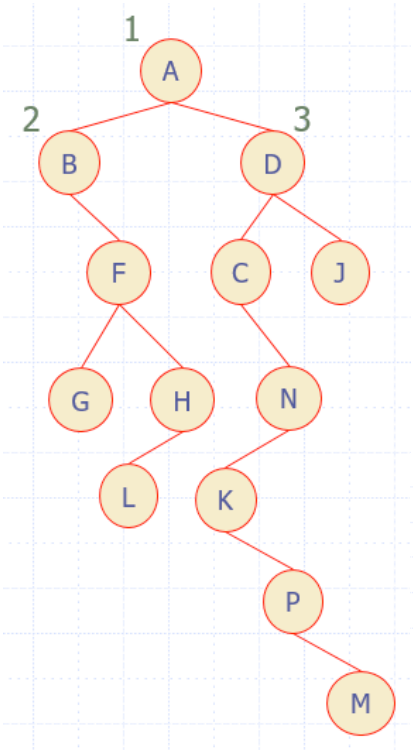
7.prove that ne=ni+1

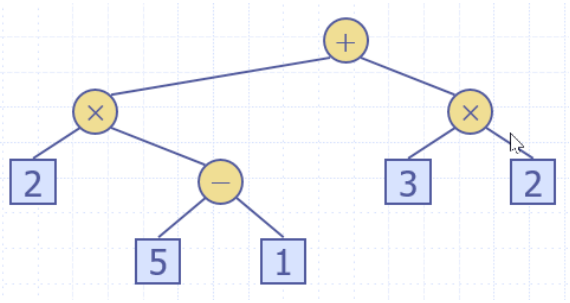
8.Given a proper binary tree explain why is the number of nodes always odd?

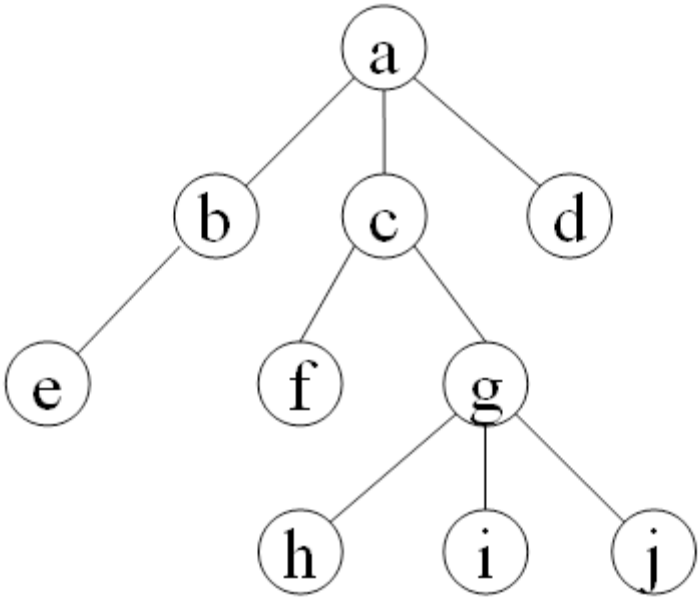
If n is 51 what is the number of internal and external nodes?

9.nk is the number of nodes with k children.

Proof that n0=n2+1



1. (2 pts) **About heaps**: Given a heap of $n$ nodes, what are its minimum and maximum values of height? Why?

2. (4 pts) **Complexity of operations on heaps**: Give the time complexity of the following operations on a min-heap of n nodes: (a) size (b) min (c) insert (d) removeMin.

3. (4 pts) **Heap construction in O(n)**: Given a vector [16 15 4 12 6 9 23 20 25 5 11 27 7 8 10], construct a min-heap from these elements based on a O(n) algorithm. Please plot the result at each step to demonstrate the algorithm clearly.

4. (4 pts) **Insertion and deletion in heaps**: Draw a max-heap tree after each of the following operations is performed, beginning with an empty

heap: insert 43, insert 31, insert 68, insert 24, delete-max, insert 51, insert 44, insert 53, insert 69, insert 71, delete-max.

5. (6 pts) **In-place heap sort**: Given a vector of [8 5 1 3 7], how do you perform in-place heap sort? Please plot the 5 heaps (both vector and tree representations) during heap construction, and the other 5 during output, as described in class.

1. (2 pts) **Two components of a hash function**: What are the two functions that are generally used to compose a hash function?

2. (2 pts) **Horner's rule**: Explain the Horner's rule for efficient polynomial evaluation of $p(z)=a0+a1z+a2z2+\cdots+an-1zn-1$.

3. (2 pts) **Basic compression function**: What is the most commonly used compression function for a hash function?

4. (2 pts) **Definition of collision**: What is the definition of collision in terms of the operation of a hash table?

5. (4 pts) **Collision handling**:
    a. What are the basic two methods for handling collision?
    b. What is the major difference betweeen the above two methods?

6. (4 pts) **Linear probing**: If we are using linear probing for handling collisions in a hash table, with a hash function h(x)=x mod $13$ Show the hash table contents after inserting keys of 18, 41, 22, 44, 59, 32, 31, 73.

7. (4 pts) **Design criteria for a hash function**: What are the most important two considerations in designing a hash function?