在答案卷上，請務必按照題號順序、子題順序作答

Problem 1 (10 points). Describe how to emulate a queue with two stacks A and B and only <u>one</u> extra variable v. We can only use the following three operations.

- Push v into a stack. For example, to push a data v into stack A is represented as 'push(v, A)'.
- Pop the top of stack element to v. For example, to pop the top of stack A to a variable v is represented as 'v = pop(A)'.
- Test if a stack is empty. For example, to test if stack A is empty is represented as 'empty(A)'.

We need to emulate two queue operations – enqueue and dequeue.

1. Enqueue is to add an element e into the tail of a queue. We implemented it as push (e, A).
2. Dequeue is to remove and return an element from the head of the queue. This is implemented as the following.
   Step 1: If ____(1a)____ {
   While ____(1b)____ do the following two things:
   ____(1c)____
   ____(1d)____
   }
   Step 2: ____(1e)____
   Step 3: Return the value of v.

Now fill in the blanks with the stack operation push, pop, and empty. We assume that we will never try to dequeue from an empty queue.

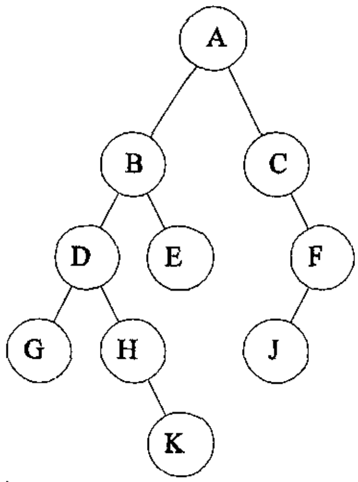
Problem 2 (10 points). The following is a recursive definition for binary tree.

1. An empty tree is a binary tree.
2. If both left and right subtrees of a tree T are binary trees, then T is also a binary tree.

Now we define an E operation on binary tree T.

1. If T is an empty binary tree, then $E(T)$ is also an empty binary tree.
2. If T is not an empty binary tree and it has left subtree L and right subtree R, then $E(T)$ has the same root as T has, and the left subtree of $E(T)$ is $E(R)$, and the right subtree of $E(T)$ is $E(L)$.

Now if T is as follows.



Please draw a picture of $E(T)$. ____(2a)____

Problem 3 (10 points). We would like to compute Pascal's triangle.

```
            1
         1     1
      1     2     1
   1     3     3     1
1     4     6     4     1
```

We now implement it in one dimensional array *B* with the following indexing. That means *B[2]* will be the first element in the second row of Pascal's triangle, and *B[9]* will be the third element in the fourth row, and so on.

```
            [1]
         [2]   [3]
      [4]   [5]   [6]
   [7]   [8]   [9]   [10]
[11] [12] [13] [14] [15]
```

Now fill in the blanks of the following pseudo code, where n is the number of rows in the triangle.

```
Set index to 1
Let i be from 1, 2, … up to n {
    Assign 1 to B[___(3a)___] and B[___(3b)___];
    Let j be from ___(3c)___ up to ___(3d)___
        Assign ___(3e)___ to B[j].
    Set index to index + i
}
```

Problem 4 (10 points; 4a, 4b: 3 points each, 4c: 4 points). We can implement a stack of integers with either an integer array or a linked list. When we implement a stack with an array, we will need an extra integer to record the location of the top of stack. When we implement a stack with a linked list, every node will have two fields - one integer and one pointer to the next node in the list. In addition, we need a pointer to point to the top of stack. Now if we know in advance that the number of integers in the stack will reach *N*, what is the memory requirements in these two implementation? We assume that a pointer requires *P* bytes, and an integer requires *I* bytes. We also assume that we use an array of *M* elements, where *M* is larger than *N*.

Now compute the memory requirement (in bytes) for array implementation __(4a)__ and for linked list implementation __(4b)__. Under what condition the linked list implementation will use more memory than the array implementation? __(4c)__

Problem 5 (10 points). We need a data structure to represent a "boss" relation. Every node in this data structure has two fields – a name and a pointer to another node. For example, if John works for Mary, then the pointer of the node representing John will point to the node representing Mary. In other words, Mary is John's boss. For ease of representation if a person does not work for anyone, we set his/her "boss" pointer to himself/herself, and call him/her a "super-boss". Since every node has a pointer, every one has at exactly one boss. As a result if we trace the boss of the boss and keep on tracing, eventually we will find a super-boss. If two persons have the same super-boss, we call them "colleagues".

Now given the following "boss" relation, draw a picture of this data structure. __(5a)__

John works for Mary
Tom works for Mary
Jimmy works Eric

接次頁

Joe works for Tom
Mary works for Kevin
Kevin works for Lucy
Joe works for Tom
George works for Tom
Barry works for Kevin
Adam works for Mary
Harry works for Tom

Also please indicate all colleagues of Mary. __(5b)__

Problem 6 (25 points). Give a tight asymptotic bound (Θ) for each T(n) in the following.

(6a) $T(n) = 3T(\frac{n}{5}) + n \lg n$

(6b) $T(0) = 0, T(1) = 1, T(n) = 5T(n-1) - 6T(n-2)$

(6c) $T(n) = \sum_{i=1}^{n} i^4$

(6d) $T(n) = \sum_{i=1}^{n} \frac{1}{i}$

(6e) $T(n) = T(\left\lceil \frac{n}{2} \right\rceil) + 1$

Problem 7 (10 points). A subsequence of a given sequence is the given sequence with some elements (possibly none) left out. For example, *ALG*, *AGTHM*, *IH* are all subsequences of the sequence "*ALGORITHM*." Given two sequences, the longest common subsequence problem is to find a subsequence that is common to both sequences and its length is maximized. Complete the following pseudocode for computing the length of a longest common subsequence of two sequences $A=a_1a_2...a_m$ and $B=b_1b_2...b_n$.

```
procedure LCS-Length(A, B)
    for i ← 0 to m do len[i,0] ← 0
    for j ← 1 to n do len[0,j] ← 0
    for i ← 1 to m do
        for j ← 1 to n do
            if a_i is equal to b_j then ___(7a)___ else ___(7b)___
    return len[m,n]
```

Problem 8 (15 points). The following procedure *Guess* takes two integers as input and returns a triple of the form (r, s, t). The call *Guess*(80,68) returns r = __(8a)__, s = __(8b)__, t = __(8c)__.

```
procedure Guess(a, b)
    if b is equal to 0 then return (a, 1, 0)
    (d, x, y) ← Guess(b, a mod b)
    (r, s, t) ← (d, y, x-⌊a/b⌋y)
    return (r, s, t)
```