1-1 In a Red-Black tree, the path from the root to the farthest leaf is no more than twice as long as the path from the root to the nearest leaf. (2分)

◉ T ○ F

1-2 While accessing a term by hashing in an inverted file index, range searches are expensive. (2分)

● T ◉ F

1-3 The weighted Activity Selection problem with weights in the set {1, 2} can be solved optimally by the same greedy algorithm used for the unweighted case. (2分)

○ T ◉ F

1-4 To solve the Maximum Finding problem with parallel Random Sampling method, $O(n)$ processors are required to get $T(n) = O(1)$ and $W(n) = O(n)$ with very high probability. (2分)

● T ◉ F

1-5 If there are 105 runs to be merged using 4 taps for 3-way merges, then distributing the runs unevenly as (17, 32, 56) will require less number of passes than the even distribution (35, 35, 35). (2分)

◉ T ● F

1-6 The set cover problem is one of Karp's 21 NP-complete problems. Given a set of elements $U = \{1, 2, ..., n\}$ (called the universe) and a collection S of k sets whose union equals the universe, and and a cost function $cost : S \rightarrow Q^+$, the weighted set cover problem is to identify the smallest cost of sub-collection of S whose union equals the universe. Consider an approximation algorithm as follows:

```
C = null
while ( C!=U ) {
    for each S[i], calculate the cost-effectiveness a[i] = cost(S[i]) / |S[i]-C|
    find the best cost-effective set, say S[j]
    pick S[j]
    C = C union S[j]
}
output the picked sets
```

If $P \neq NP$, we can't find a constant $\rho \in N^+$, so that the approximation ratio of the above algorithm is ρ.

(2分)

● T ◉ F

1-7 If Φ is a potential function associated with a data structure S, then 3Φ is also a potential function that can be associated with S. Moreover, the amortized running time of each operation with respect to 3Φ is at most triple the amortized running time of the operation with respect to Φ. (2分)

◉ T ○ F

1-8 Local search algorithm can be used to solve lots of classic problems, such as SAT and N-Queen problems. Define the configuration of SAT to be X = vector of assignments of N boolean variables, and that of N-Queen to be Y = positions of the N queens in each column. The sizes of the search spaces of SAT and N-Queen are $O(2^N)$ and $O(N^N)$, respectively. (2分)

◉ T ○ F

1-9 If devide-and-conquer strategy is used to find the closest pair of points in a plane, unless the points are sorted not only by their x coordinates but also by their y coordinates, it would be impossible to solve it in a time of $O(NlogN)$, where N is the number of points. (2分)

◉ T ○ F

1-10 A boolean formula is in **disjunctive normal form** (DNF) if it is a disjunction (OR) of one or more conjunctions (AND) of one or more literals. A literal is a variable or its negation. For example, the formula (x ∧ ¬y ∧ z) ∨ (¬x ∧ z) ∨ (w ∧ y ∧ ¬z) is a DNF formula. In DNF-Sat you are given a DNF formula, and the goal is to tell whether that formula is satisfiable.

We claim that the problem DNF-Sat is in P. (2分)

● T ◉ F

1-11 In the Activity Selection problem, consider any non-empty set of activities S, and let a_m be an activity in S with the ***shortest lasting time***. Then a_m must be included in some maximum-size subset of mutually compatible activities of S. (2分)

○ T ◉ F

1-12 A leftist heap with the null path length of the root being r must have at least $2^{r+1}-1$ nodes. (2分)

◉ T ○ F

1-13 Inserting a number into a binomial heap with 11 nodes costs more time than inserting a number into a binomial heap with 17 nodes. (2分)

◉ T ○ F

1-14 Reviewing the randomized QuickSort in our course, we always select a central splitter as a pivot before recursions, make sure that each side contains at least $n/4$ elements. Hence, differing from the deterministic QuickSort, the **worst case expected running time** of the randomized QuickSort is $\Theta(NlogN)$. (2分)

◉ T ○ F

1-15 In a Turnpike Reconstruction problem, given the distance set D = { 1, 2, 2, 3, 3, 4, 5, 6, 6, 8 }, it is impossible to have a point placed at 5. (2分)

○ T ◉ F

2-1 Given a linked list containg N nodes. Our task is to remove all the nodes. At each step, we randomly choose one node in the current list, then delete the selected node together with all the nodes after it. Here we assume that each time we choose one node uniformly among all the remaining nodes. What is the expected number of steps to remove all the nodes? (3分)

- (●) A. $\Theta(logN)$
- ( ) B. N/e
- ( ) C. $N/2$
- ( ) D. $\sqrt{N}$

2-2 When solving a problem with input size N by divide and conquer, if at each step, the problem is divided into 4 sub-problems of size $\sqrt{N}$, and they are conquered in $O(logN)$. Which one of the following is the closest to the overall time complexity $T(N)$? Suppose that $T(2) = O(1)$ and all related root values are integers. (3分)

- ( ) A. $O(logN)$
- (●) B. $O(log^2N)$
- ( ) C. $O(N)$
- ( ) D. $O(loglogN)$

2-3 In a binomial queue with 150 nodes, how many nodes have depth 1 (the root has depth 0)? (3分)

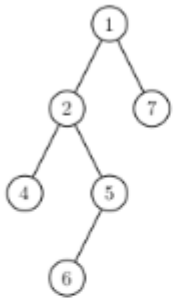
- ( ) A. 4
- ( ) B. 8
- (●) C. 14
- ( ) D. Cannot be determined

2-4 Suppose that the devide-and-conquer strategy is used to find the maximum and the minimum of N positive numbers. At each step, the problem is divided into 2 sub-problems of size $N/2$. Then the time recurrences is $T(N) = 2T(N/2) + f(N)$, where $f(N)$ is __. (3分)

- ( ) A. $\Omega(N)$
- (●) B. $O(1)$
- ( ) C. $N/2$
- ( ) D. $\Theta(logN)$

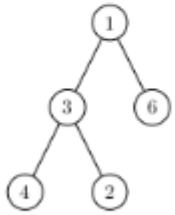
2-5 Which one of the following statements is **TRUE**? (3分)

A.



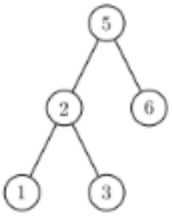
may be a leftist heap

B.



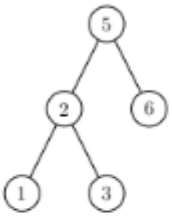
may be a leftist heap

C.



may be a leftist heap

D.



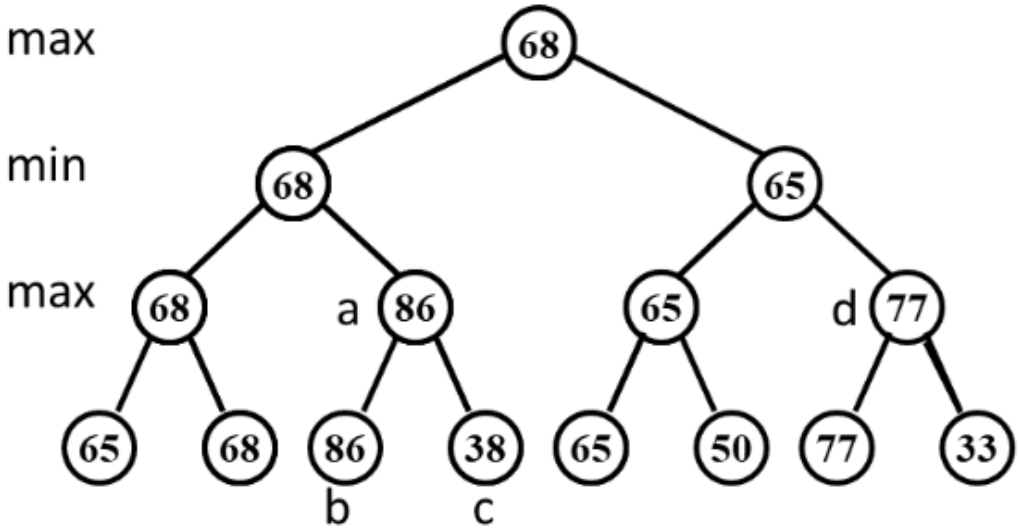
may be a skew heap

2-6 How many of the following statements is/are **TRUE**? (3分)

- The 0-1 knapsack problem cannot be solved by any local search algorithm.
- The metropolis algorithm always improves the gradient descent algorithm.
- In some cases, the state-flipping algorithm cannot terminate.
- Unless $P = NP$, there is no ρ-approximation for the maximum cut problem for any $\rho < 2$.

A. 0
B. 1
C. 2
D. 3

2-7 Given the following game tree, which node is the first one to be pruned with α-β pruning algorithm? (3分)



- A. a
- B. b
- C. c
- D. d

2-8 The potential of a skew heap is defined to be the total number of right heavy nodes. The weight of a node, $w(x)$, is defined to be the number of descendants of x (including x). A non-root node is said to be **heavy** if its weight is greater than half the weight of its parent.

Lemma 1: At most one child is heavy, of all children of any node.

Lemma 2: On any path from node x down to a descendant y, there are at most $\lfloor log_2 \frac{w(x)}{w(y)} \rfloor$ light nodes, excluding x. In particular, any path in an n-node tree contains at most $\lfloor log_2 n \rfloor$ light nodes.

Define the following functions:

- *makeheap*: create a new, empty heap
- *findmin*: return the minimum key in the heap
- *insert*: insert a key in the heap
- *deletemin*: delete the minimum key from the heap
- *merge*: merge two heaps

Then for any n-node skew heaps, which of the following is **FALSE**? (3分)

- A. The amortized time of a merge operation is $O(logn)$
- B. The amortized time of a makeheap operation is $O(1)$
- C. The amortized time of a findmin is $O(1)$
- D. The amortized time of a deletemin operation is $O(1)$

2-9 Let X be a problem that belongs to the class NP. Then which one of the following is **TRUE**? (3分)

- A. There is no polynomial time algorithm for X.
- B. If X can be solved deterministically in polynomial time, then P = NP.
- C. If X is NP-hard, then it is NP-complete.
- D. X may be undecidable.

2-10 Delete the minimum number from the binomial queue given in the following figure. Which one of the following statements must be **FALSE**? (3分)

- A. there are two binomial trees after deletion, which are B_2 and B_3
- B. 23 and 14 are both the roots of some binomial trees
- C. 21 is the child of 13
- D. 23 and 24 are siblings

2-11 Suppose that the replacement selection is applied to generate longer runs with a priority queue of size 4. Given the sequence of numbers { 9, 75, 17, 12, 88, 91, 25, 22, 35, 41, 58, 96, 15 }. Which of the following gives the second output run? (3分)

- A. 22 35 88 91
- B. 22 35 41 58 88 91 96
- C. 22 35 41 58 96
- D. 22 35 41 58

2-12 When doing amortized analysis, which one of the following statements is FALSE? (3分)

- A. Aggregate analysis shows that for all n, a sequence of n operations takes worst-case time $T(n)$ in total. Then the amortized cost per operation is therefore $T(n)/n$
- B. For potential method, a good potential function should always assume its maximum at the start of the sequence
- C. For accounting method, when an operation's amortized cost exceeds its actual cost, we save the difference as credit to pay for later operations whose amortized cost is less than their actual cost
- D. The difference between aggregate analysis and accounting method is that the later one assumes that the amortized costs of the operations may differ from each other

2-13 Insert { 5, 1, 7, 8, 21, 2, 12, 19, 13, 0 } into an initially empty 2-3 tree (with splitting). Which one of the following statements is FALSE? (3分)

- A. 13 and 19 are in the same node
- B. the parent of the node containing 8 has 3 children
- C. the first key stored in the root is 7
- D. there are 5 leaf nodes

2-14 To delete X from a splay tree, the operations are: (1) Find X; (2) Remove X; (3) FindMax(T_L); and the last operation is: (3分)

- A. Make T_L the right child of the root of T_R
- B. Make T_L the left child of the root of T_R
- C. Make T_R the right child of the root of T_L
- D. Make T_R the left child of the root of T_L

2-15 There are 8000 documents in the database. The statistic data for one query are shown in the following table. The precision is: __ (3分)

	Relevant	Irrelevant
Retrieved	1000	1000
Not Retrieved	2000	4000

- A. 12.5%
- B. 20%
- C. 33%
- D. 50%

2-16 Which of the following is **TRUE** about NP-Complete and NP-Hard problems? (3分)

- A. If we want to prove that a problem X is NP-Hard, we take a known NP-Hard problem Y and reduce Y to X in polynomial time.
- B. The first problem that was proved to be NP-complete was the circuit satisfiability problem.
- C. NP-complete is a subset of NP-Hard
- D. All of the other three.

2-17 Given 4 cases of frequences of four characters. In which case(s) that the total bits taken by Huffman codes are the same as that of the ordinary equal length codes? (3分)

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

- A. (1) and (2)
- B. (3) only
- C. (1), (2) and (4)
- D. (2) only

2-18 Assume that you are a real world Chinese postman, which have learned an awesome course "Advanced Data Structures and Algorithm Analysis" (ADS). Given a 2-dimensional map indicating N positions $p_i(x_i, y_i)$ of your post office and all the addresses you must visit, you'd like to find a shortest path starting and finishing both at your post office, and visit all the addresses at least once in the circuit. Fortunately, you have a magic item "Bamboo copter & Hopter" from "Doraemon", which makes sure that you can fly between two positions using the directed distance (or displacement).

("Bamboo copter & Hopter", japan12.com/bamboo-copter-hopter)

However, reviewing the knowledge in the ADS course, it is an NPC problem! Wasting too much time in finding the shortest path is unwise, so you decide to design a $2-approximation$ algorithm as follows, to achieve an acceptable solution.

```
Compute a minimum spanning tree T connecting all the addresses.
Regard the post office as the root of T.
Start at the post office.
Visit the addresses in order of a _____ of T.
Finish at the post office.
```

There are several methods of traversal which can be filled in the blank of the above algorithm. Assume that $P \neq NP$, how many methods of traversal listed below can fulfill the requirement? (3分)

- Level-Order Traversal
- Pre-Order Traversal
- Post-Order Traversal

A. 0
B. 1
C. 2
D. 3

2-19 Which one of the following problems can be best solved by dynamic programming? (3分)

A. Mergesort
B. Closest pair of points problem
C. Quicksort
D. Longest common subsequence problem

2-20 The following psuedo-code is for solving the Profix Sums problem parallely, where the input N numbers are stored in the array `A`. Which of the following gives the time and work load of the algorithm? (3分)

```
for Pi , 1 <= i <= n pardo
    B(0, i) := A(i)
for h = 1 to log(n)
    for i, 1 <= i<= n/(2^h) pardo
        B(h, i) := B(h-1, 2*i-1) + B(h-1, 2*i)
for h = log(n) to 0
    for i even, 1 <= i<= n/(2^h) pardo
        C(h, i) := C(h+1, i/2)
    for i = 1 pardo
        C(h, 1) := B(h, 1)
    for i odd, 3 <= i <= n/(2^h) pardo
        C(h, i) := C(h+1, (i-1)/2) + B(h, i)
for Pi , 1 <= i<= n pardo
    Output C(0, i)
```

A. $T(n) = O(logn), W(n) = O(nlogn)$
B. $T(n) = O(n), W(n) = O(n)$
C. $T(n) = O(logn), W(n) = O(n)$
D. $T(n) = O(n), W(n) = O(nlogn)$

5-1 IsRBTree (3)

The functions `IsRBTree` is to check if a given binary search tree `T` is a red-black tree. Return `true` if `T` is, or `false` if not.

The red-black tree structure is defined as the following:

```
typedef enum { red, black } colors;
typedef struct RBNode *PtrToRBNode;
struct RBNode{
    int Data;
    PtrToRBNode Left, Right, Parent;
    int BH; /* black height */
    colors Color;
};
typedef PtrToRBNode RBTree;
```

Please fill in the blanks.

```
bool IsRBTree( RBTree T )
{
    int LeftBH, RightBH;
    if ( !T ) return true;
    if ( T->Color == black ) T->BH = 1;
    else {
         if ( T->Left && (T->Left->Color == red)) return false;
         if ( T->Right && (T->Right->Color == red)          (2分) ) return false;
    }
    if ( !T->Left && !T->Right ) return true;
    if ( IsRBTree(T->Left) && IsRBTree(T->Right)          (2分)) {
       if ( T->Left ) LeftBH = T->Left->BH;
       else LeftBH = 0;
       if ( T->Right ) RightBH = T->Right->BH;
       else RightBH = 0;
       if ( LeftBH == RightBH ) {
          T->BH += LeftBH                                 (2分);
          return true;
       }
       else return false;
    }
    else return false;
}
```

You are supposed to implement the `Insert` function, which inserts an integer `Key` into an AVL tree `T`. The resulting tree must be returned.

Format of function:

```
AVLTree Insert ( AVLTree T, int Key );
```

where `AVLTree` is defined as the following:

```
typedef struct AVLNode *PtrToAVLNode;
struct AVLNode{
    int Key;
    PtrToAVLNode Left;
    PtrToAVLNode Right;
    int Height;
};
typedef PtrToAVLNode AVLTree;
```

Sample program of judge:

```
#include <stdio.h>
```

查看上次提交 提交

```
struct AVLNode{
    int Key;
    PtrToAVLNode Left;
    PtrToAVLNode Right;
    int Height;
};
typedef PtrToAVLNode AVLTree;

AVLTree Insert ( AVLTree T, int Key );
void PostOrderPrint( AVLTree T ); /* details omitted */
void InOrderPrint( AVLTree T );   /* details omitted */

int main()
{
    int N, Key, i;
    AVLTree T = NULL;

    scanf("%d", &N);
    for ( i=0; i<N; i++ ) {
        scanf("%d", &Key);
        T = Insert( T, Key );
    }
    PostOrderPrint( T );
    InOrderPrint( T );

    return 0;
}
/* Your function will be put here */
```

Sample Input:

查看上次提交 提交

Sample Output:

```
Post-order: 61 70 65 90 120 96 88
In-order: 61 65 70 88 90 96 120
```