1. Company ABC is conducting an interview for the post of manager. The candidates are shortlisted for the interview after a written test. Each candidate is given a unique rank (starting from 1) based on their marks in the test such that the highest rank (rank 1) is given to the candidate with highest mark.

   Interview starts at time s. For each candidate, a time slot of m minutes is given.

   The order in which the candidates are called for interview is as follows: At each time t, from the candidates arrived before t, the candidate with the highest rank (lowest number) is selected for the interview. After the interview of a candidate is over, the same procedure is repeated for selecting the next candidate for the interview.

   Assumptions:

   - At least one candidate has arrived before the starting time s.
   - Interviews of all the candidates finish in the same day.
   - There is no break in between the interviews.

   Note: **Follow 24 hour format.**

   Given the arrival time of n candidates shortlisted for the interview and their ranks (sorted in arrival time), write a C program to print the interview schedule.

Input Format

- The first line contains the number of candidates shortlisted for the interview n, the starting time of the interview s and the time slot $m \in [1, 60]$, separated by a space.
- Each of the next n lines contains the arrival time a, candidate id i and the rank r of each of the candidates, separated by a space.

The integers n, i, and r are in the range $[1, 10^3]$. The starting time s and the arrival time a of each candidate is given as floating point numbers.

Output Format

- Print the interview schedule as given in the sample output.

Sample Input and Output

Input 1

```
7 9.00 10
8.30 104 4
8.35 105 5
8.50 103 3
9.05 106 6
9.20 101 1
9.35 102 2
9.43 107 7
```

Output 1

```
9.00 103
9.10 104
9.20 105
9.30 101
9.40 102
9.50 106
10.00 107
```

Input 2

```
5 11.30 30
10.30 104 4
10.25 105 5
11.50 103 3
12.15 106 6
12.30 101 1
```

Output 2

```
11.30 104
12.00 103
12.30 105
13.00 101
13.30 106
```

General Instructions:

- If required, you may use global variable(s) for this question.

---

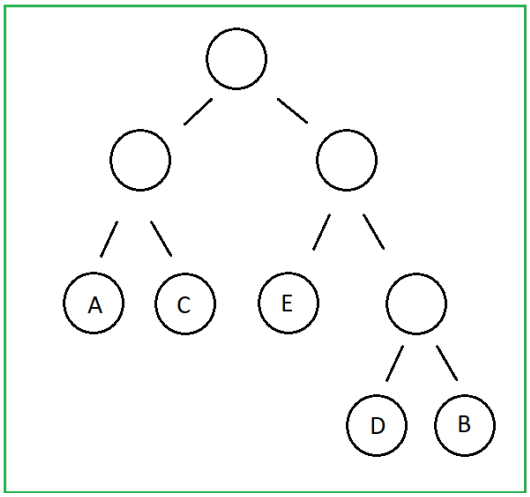
2. An *encoding tree* T is a binary tree in which the leaves contain alphabets and the path from the root of the tree to the leaf containing an alphabet defines the encoding of that alphabet.

   The code for an alphabet in T is obtained as,

   - Traverse T from root to the leaf containing the alphabet.
   - While traversing T from root, at each node if the alphabet is to the left of the current node, add 0 to the code. If the alphabet is to the right of the current node, add 1 to the code.

   Example:

   Consider the given tree showing the encoding of 5 alphabets.



   The code generated for the alphabets will be:

```
A - 00
B - 111
C - 01
D - 110
E - 10
```

   Given an encoding tree T as its parenthesis representation, write a C program to generate the codes for the alphabets in T.

Input Format

- A single line containing the parenthesis representation (maximum length 1000) of the encoding tree T. Each of the leaf nodes is represented by an alphabet $\in [A - Z]$. The internal nodes are represented using a dummy character $*$.

Output Format

- Print the codes generated, in alphabetical order as given in the sample output.

Sample Input and Output

Input 1

```
( * ( * ( A ( ) ( ) ) ( C ( ) ( ) ) ) ( * ( E ( ) ( ) ) ( * (
D ( ) ( ) ) ( B ( ) ( ) ) ) ) )
```

Output 1

```
A 00
B 111
C 01
D 110
E 10
```

Input 2

```
( * ( * ( P ( ) ( ) ) ( N ( ) ( ) ) ) ( * ( E ( ) ( ) ) ( * (
) ( F ( ) ( ) ) ) ) )
```

Output 2

```
E 10
F 111
N 01
P 00
```

General Instructions:

- Your implementation should strictly follow all the instructions given in this question.
- The implementation will be evaluated manually, and if it does not follow the instructions given in the question, **NO marks will be awarded even if the implementation passes all the test cases**.

---

3. You are given a collection of n Binary Search Trees. Your objective is to *order* these trees.

   Definition 1. *The height of a binary tree T, $height(T)$ is defined as the number of edges on the longest simple downward path from the root of T to a leaf. If T has only one node (the root), then $height(T) = 0$.*

   Definition 2. *A collection of n Binary Search Trees $T_1, T_2, ..., T_n$ are said to be ordered if, for all $1 \leq i < j \leq n$, $height(T_i) \leq height(T_j)$.*

   Insert the trees one by one into a Singly Linked List L. After each insertion, the trees in L should remain *ordered* (The head of L points to the shortest tree). If two trees are having the same height, they should be present in L in the order they were given in the input.

   Write a C program to insert the given Binary Search Trees into a Singly Linked List L as specified above. Use the below given structures for implementing the Binary Search Tree and Singly Linked List.

```
struct treeNode{
    int key;
    struct treeNode* left;
    struct treeNode* right;
};

struct BinarySearchTree{
    struct treeNode* root;
    int height;
};

struct listNode{
    struct BinarySearchTree* tree;
    struct listNode* next;
};

struct LinkedList{
    struct listNode* head;
};
```

Your program should include the following functions as per the function prototypes given below.

- $treeInsert(T, k)$: Insert k into the Binary Search Tree T.
- $treePrint(T)$: Print the post-order traversal of the Binary Search Tree T.
- $listInsert(L, T)$: Insert the Binary Search Tree T into the Singly Linked List L.
- $listPrint(L)$: Print the post-order traversal of the ordered trees stored in the Singly Linked List L (in separate lines), starting from the head.

Input Format

- The first line contains an integer $n \in [1, 1000]$ representing the number of Binary Search Trees.
- Each of the subsequent n lines contains the key values $\in [-1000, 1000]$ in the order in which they should be inserted into each of the Binary Search Trees.

Output Format

- The post-order traversal of the *ordered* trees stored in the linked list (in separate lines), starting from the head.

Sample Input and Output

Input 1

```
6
20 23 13 5 89 14
40
56 34 23 76 45 24
1 2 3 4 5 6
23 5 45 34 22 90 4 33 27 98 76
10 9 8 5
```

Output 1

```
40
5 14 13 89 23 20
24 23 45 34 76 56
5 8 9 10
4 22 5 27 33 34 76 98 90 45 23
6 5 4 3 2 1
```

Input 2

```
4
-20 -23 13 5 89 -14
40 20 50
56 -34 23 76 45 -24
-1 -2 -3
```

Output 2

```
20 50 40
-3 -2 -1
-23 -14 5 89 13 -20
-24 45 23 -34 76 56
```