CS 602
Algorithm Design and Implementation

Assignment 2

[Question 1] Transform a Binary Tree to a Binary Search Tree

In this question, you are asked to convert an arbitrary binary tree to a binary search tree with the **same structure** and the same set of keys. Let the binary tree be T, your task is to construct a binary search tree T' such that for any pair of node n and n' at the same position (same path from the root) in T and T' respectively, the number of nodes in the left subtree of n is the same as that in the left subtree of n', and the number of nodes in the right subtree of n is also the same as that in the right subtree of n'. In addition, binary tree T' satisfies the property of binary search trees.

Implement a function to transform an arbitrary binary tree to a binary search tree. Test inputs begin with the number of lines below. Each line describes a binary tree by a list of tree nodes with key at itself, its left child and its right child, separated by '`:`'. Letter `x` represents a null node, e.g., a tree node with two `x` is a leaf node. The sequence of tree nodes is arbitrary. For example, '`0:x:x -1:1:-2 -2:0:x 1:x:x`' represents a 4-node binary tree rooted at `-1`, the left child is `1` and the right child is `-2`, where node `1` is a leaf node. For each binary tree, output the binary search tree in **pre-order traversal** separated by space, '`-1 -2 1 0`' for the above example.

Sample code is provided in the file **A2Q1.py**. You need to modify the function `to_BST`. The maximum of tree height is 100 and the maximum number of nodes in a binary tree is 1,000,000. Keys are integers between -2^{31} and $2^{31}-1$, and they are distinct.

There will be 8 sets of test cases with 1 mark each. The remaining 2 marks are awarded with the correct justification of the time complexity of your algorithm.

Sample Input	Sample Output
Please refer to the file A2Q1.in .	Please refer to the file A2Q1.out .

[Question 2] Project Selection

As the chief project manager of the team, one of your responsibilities is to select a set of projects for the team. Each project *i* is associated to a cost c_i and a revenue r_i, where c_i represents the expenditure required to work on project *i* and r_i is the revenue generated upon its completion. Your team starts with a capital value C and can only select one at a time from projects where the cost does not exceed the team's current capital. The team's capital increases by the profit of each project selected, which is r_i-c_i for project *i*. You may also assume projects are all valid with $r_i \geq c_i \geq 1$.

Given the set of projects, an initial capital value C, and k the number of projects to be selected, you are to work out a selection plan such that the final capital is maximized.

Implement an algorithm using an appropriate data structure. You may use external libraries without having to implement this data structure yourself. To test if a library (e.g., xyz) is available, perform a mock submission on the online judge by adding the line "import xyz" to the code skeleton. If you receive a "wrong answer" message, it means the library is available and can be used in your implementation.

Please also state and justify the time complexity of your algorithm.

Test inputs begin with the number of projects n ($n \leq 100,000$) and the number of scenarios. The second line contains a list of integer pairs of cost and revenue separated by ':'. These pairs are all distinct, it is possible for two projects to have the same cost or the same revenue, but not both. From the third line onward, each line describes one scenario with initial capital C and the number of projects to be selected k, where $k \leq n$. Both the capital and the costs of projects are integers and ranging between 1 and 1,000,000.

There will be 8 sets of test cases with 1 mark each. The remaining 2 marks are awarded with the correct justification of the time complexity of your algorithm.

Sample Input

```
10 3
330:510 9:12 733173:864046 664:907 646346:764956 16006:21763 9804:11465
<continued from the previous line> 14655:23652 37023:44290 18:22
9931 5
29718 7
33 5
```

Sample Output

```
12022
53827
impossible
```

[Question 3] Grouping students in a student care center

Dynamic Programming. Students typically go to student care centers after school. These centers will then partition students into groups and supervise them for self-study. However, there may be pairs of twins from the same family who always talk to each other, center staffs will not want to place them in the same group. Suppose there are n students including m pairs of twins, how many ways can the student care center partition these n students into k groups? It is allowed to have only one student in a group, but not allowed to have an empty group. The order of the groups does not matter, so it is considered as the same way if one grouping can be converted to another by swapping the sequence of the groups.

Inputs begin with the number of lines below. Each line contains three integers: n ($3 \le n \le 50$), m ($0 \le m \le 5$) and k ($3 \le k \le 20$). It is guaranteed that $2 \times m \le n$. For each line of input, output one integer which is the number of ways to partition n students into k groups without placing any of the m pairs of twins in the same group.

4 marks are award for the correct formulation and implementation of the recurrence relation (implementation of a recurrence relation is the recursion in code) to compute the result for a triple (n, m, k). Another 4 marks are awarded to 4 testcases with 1 mark each. The remaining 2 marks are awarded with the correct justification of the time complexity of your algorithm.

Sample Input

```
3
4 0 3
5 1 3
6 2 4
```

Sample Output

```
6
19
46
```

[Hint: You may refer to Stirling numbers of the second kind.]

Running python skeleton with sample input:

1. Open “Anaconda Prompt”
2. Go to the directory where you put the file **A2Q1.py** and **A2Q1.in**, using command **cd**
3. Run command **python A2Q1.py < A2Q1.in**
4. You may want to create a test input called **my_own_test.in** to design a test case for your own program, the command would then be **python A2Q1.py < my_own_test.in**
5. Same applies to Question 2, so you may run **python A2Q2.py < A2Q2.in**