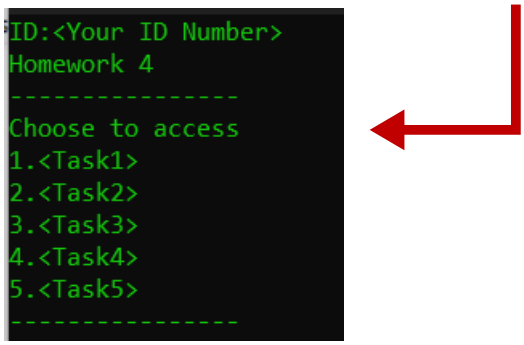
#Homework 4 (Linked list & Tree)

Rules:

- ✓ Deadline for completion of homework until: 19 May 2021 (23:59, *before midnight*)
- ✓ 20 points will be deducted for one day delay, you will get 0 points if you are 5 days late.
- ✓ Upload your homework on Portal:
  - Compress your homework folder into .rar / .zip / .7z
  - Use your student id as the name for your homework, along with the homework code.(Example: 1086412_HW4.rar / 1086412_HW4.zip / 1086412_HW4.7z).

  (important)
- ✓ C++ allowed library <iostream>, <fstream>,<cmath>,<cstdlib>, <string>, <vector>, <queue> do not use other kind of library.
- ✓ Use the Homework4_main.cpp that already provided to combine all of your codes.



Homework Tasks:

➢ **Task 1 (20%)**

Create a program to move **odd numbers** to the first of the **linked list**, with any kind of order.

Example:

Input:

1,2,3,4,5,6,7

Output:

1,3,5,7,2,4,6

Odd Numbers

- **Task 2 (20%)**

  You are asked to cut off all the trees in a forest for a golf event. The forest is represented as an m x n matrix. In this matrix:

  - 0 means the cell cannot be walked through.
  - 1 represents an empty cell that can be walked through.
  - A number greater than 1 represents a tree in a cell that can be walked through, and this number is the tree's height.

  In one step, you can walk in any of the four directions: north, east, south, and west. If you are standing in a cell with a tree, you can choose whether to cut it off.

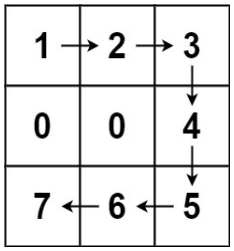
  You must cut off the trees in order from shortest to tallest. When you cut off a tree, the value at its cell becomes 1 (an empty cell).

  Starting from the point (0, 0), **return the minimum steps** you need to walk to cut off all the trees. If you cannot cut off all the trees, return -1.

  You are guaranteed that **no two trees have the same height**, and there is at least one tree needs to be cut off.

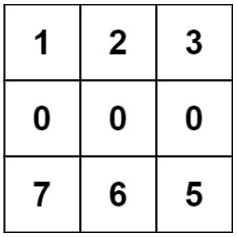
Example:

Input:
```
1,2,3
0,0,4
7,6,5
```
Output:
```
6
1->2->3->4->5->6->7
```



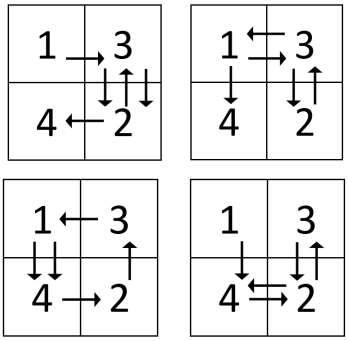
Example:

Input:
```
1,2,3
0,0,0
7,6,5
```
Output:
```
-1
```



Example:

Input:
```
1,3
4,2
```
Output:
```
5
1->3->2->3->2->4
1->3->2->3->1->4
1->4->2->3->1->4
1->4->2->3->2->4
```



- **Task 3 (20%)**

  Create a program to efficiently merge `k` **sorted linked lists**, from listed sorted linked list.

  - The number of every list cannot be the same, it must be show error
  - The `k` linked list order is from the first input to the last input.
  - The total of list limit is 10,
  - The total of number inside limit is 10.

Example:

```
Input:
2
1,3,5
2,4,6



1 -> 2 -> 3 -> 4 -> 5 -> 6
```

Example:

```
Input:
3
1,5
2,4
8,9
3,6
7,10

Output:
1 -> 2 -> 4 -> 5 -> 8 -> 9
```

Example:

```
Input:
5
1,5
2,4
8,9
3,6
7,10


1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 7 -> 8 -> 9 -> 10
```

➢ **Task 4 (20%)**

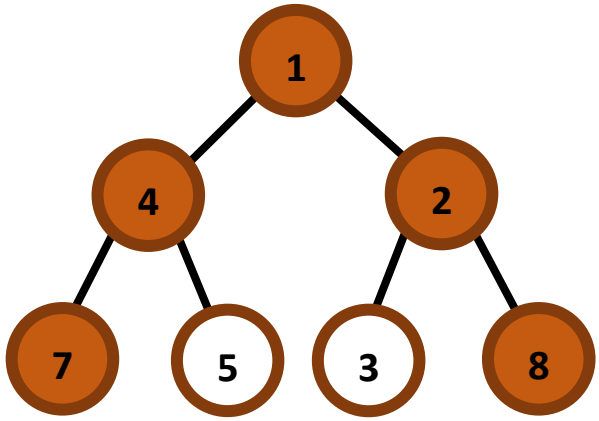
Create a program to find maximum path sum in a structured binary tree :

- Remember this is binary tree, the leaf or child **cannot more than two nodes**.
- Minimal connection is one between two vertices.
- Start and stop vertex can be anywhere as long the connection is the maximum path sum.
- If there any connection with the maximum path have the same maximum paths, it doesn't matter to chose which one, because the output just the maximum path sum

Example:

Input:
`[1,4],[1,2],[4,7],[4,5],[2,3],[2,8]`

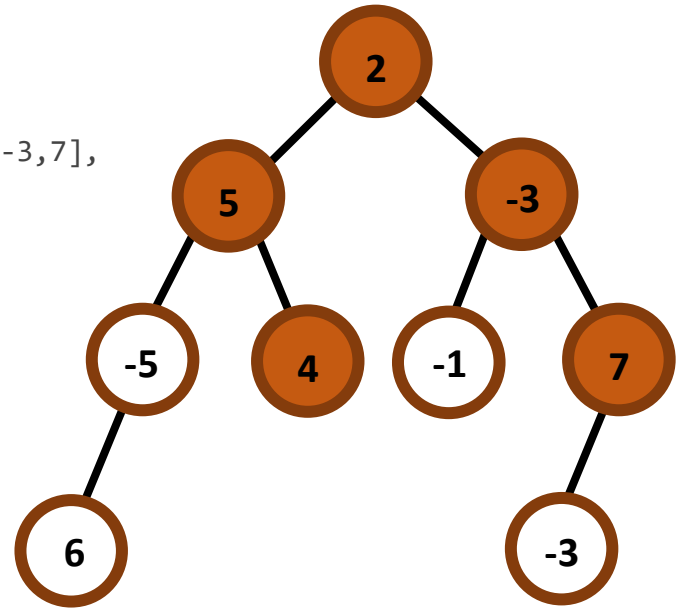
Output:
`The maximum path sum is 22`



Example:

Input:
`[2,5],[2,-3],[5,-5],[5,4],[-3,-1],[-3,7],`
`[-5,6],[7,-3]`

Output:
`The maximum path sum is 15`

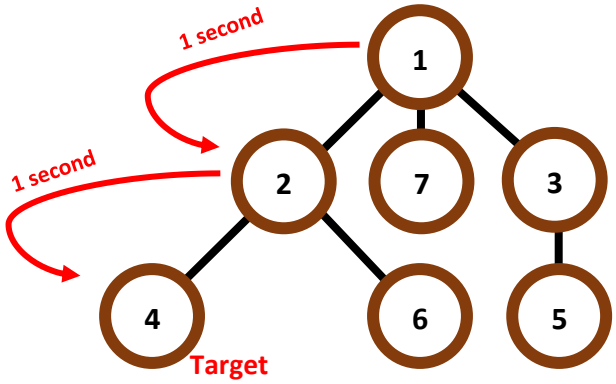


➢ **Task 5 (20%)**

- Given an undirected tree consisting of n vertices numbered from 1 to n. A monkey starts jumping from vertex 1.
- **In one second**, the monkey jumps from its **current vertex to another unvisited vertex** if they are directly connected.
- The monkey **cannot jump back to a visited vertex**.
- Otherwise, when the monkey cannot jump to any unvisited vertex, it jumps forever on the same vertex.
- The edges of the undirected tree are given in the array edges, where edges [i] = [ai, bi] means that exists an edge connecting the vertices ai and bi.
- Return the **probability** that after t seconds the monkey is on the vertex target. Answers within 10^{-5} or 0.00001 of the actual answer will be accepted.
- If the time is not enough to jump until the target vertex, set the output **"Not Enough Time".**

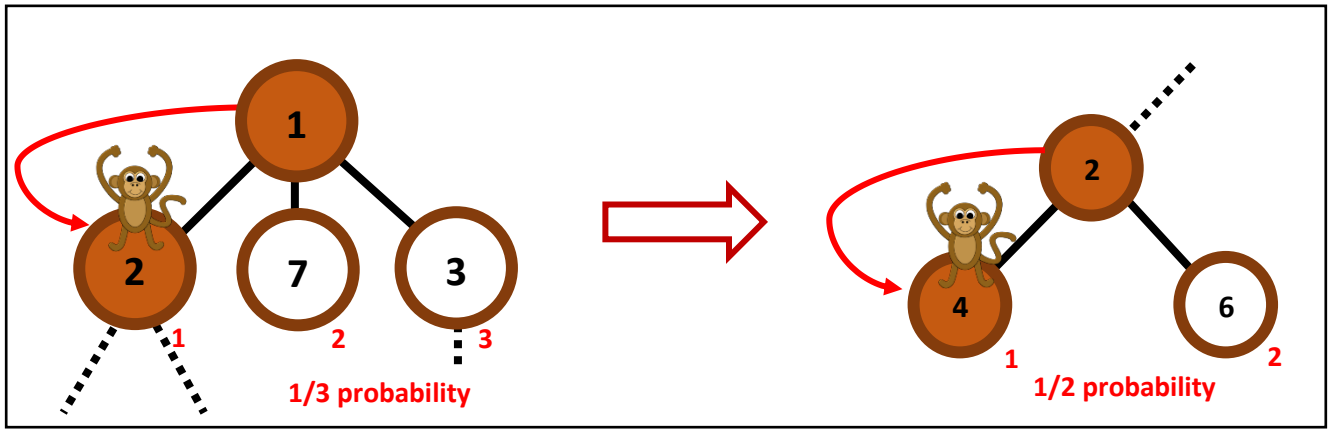
```
Input:
t = 2          Time limit (second)
target = 4
[1,2],[1,3],
[1,7], [2,4],
[2,6],[3,5]

Output:
0.16666666
```



How to get output probability?

From vertex 1 to 2 = 1/3
From vertex 2 to 4 = 1/2
Then,
From vertex 1 to 4 = 1/3 * 1/2 = 0.16666666

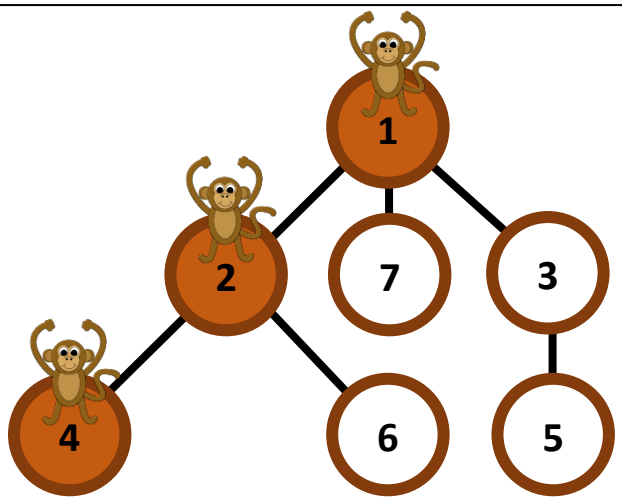


Example:

Input:

2 **Time limit (second)**
4 **Target**
[1,2],[1,3],[1,7],[2,4],[2,6],[3,5]

Output:
0.16666666666666666

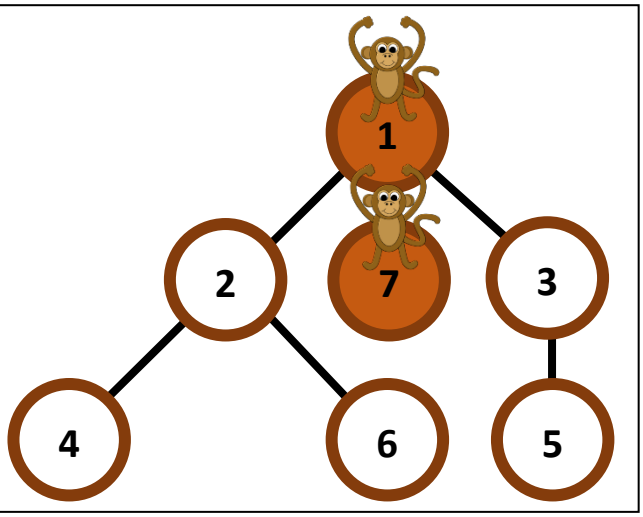


Example:

Input:

1 **Time limit (second)**
7 **Target**
[1,2],[1,3],[1,7],[2,4],[2,6],[3,5]

Output:
0.3333333333333333



TA Information:

Name: **Gideon (吉迪恩)**

Email: dywithly@gmail.com

Lab Room: **Building 5 – Room 5402**