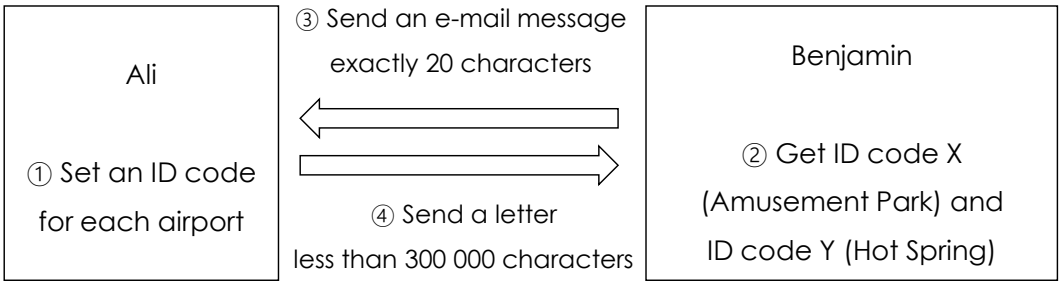
Flights

In Republic of JOI, there are N airports numbered from 0 to $N-1$. There are $N-1$ airline routes numbered from 0 to $N-2$. The airline route i ($0 \leq i \leq N-2$) connects the airport U_i and the airport V_i bidirectionally. It is possible to travel from any airport to any other airport by connecting several airline routes. For every airport, there are at most 3 airline routes connecting it with another airport.

Benjamin is planning to take a trip in Republic of JOI. On the last day of the trip, he wants to arrive at the airport where the hot spring is located. The amusement park is located at the airport x, and the hot spring is located at the airport y. Since Benjamin does not know anything about the airline routes, he will communicate with Ali, a staff of the airplane company. Benjamin wants to know the minimum number of airline routes he has to take to travel from the airport where the amusement park is located to the airport where the hot spring is located. Ali knows information of the airplane routes. But Benjamin does not know which airline routes he has to take.

1. Ali sets an **ID code** for each airport. An ID code is an integer between 0 and $2N + 19$, inclusive.
2. Benjamin gets the ID code X of the airport where the amusement park is located, and the ID code Y of the airport where the hot spring is located.
3. Benjamin sends an e-mail message to Ali. The message is a string whose length is **exactly** equal to 20. Every character of the message is either 0 or 1.
4. Ali writes a letter to Benjamin. The letter contains a string whose length is between 1 and 300 000, inclusive. Every character of the letter is either 0 or 1.

Write programs which implement the strategy of Ali, a staff of the airplane company, and the strategy of Benjamin, a traveler. Note that in Step 2, Benjamin can get the ID codes X, Y of the airports where the amusement park and the hot spring are located. **However, Benjamin cannot get the airport numbers x, y.**



Implementation Details

You need to submit two files.

The first file is `Ali.cpp`. It should implement Ali's strategy. It should implement the following two functions. The program should include `Ali.h` using the preprocessing directive `#include`.

- `void Init(int N, std::vector<int> U, std::vector<int> V)`
  This function implements Ali's strategy to set ID codes for each airport. For each scenario (see Grading for details), this function is called exactly once.
  - The parameter `N` is the number of airports in Republic of JOI.
  - The parameters `U`, `V` are arrays of length $N-1$. This means `U[i]`, `V[i]` are the airports U_i, V_i connected by the airline route i ($0 \leq i \leq N-2$).
- `std::string SendA(std::string S)`
  This function implements Ali's strategy to send a letter to Benjamin. For each scenario (see Grading for details), this function is called exactly once after the function `SendB` (see below) is called.
  - The parameter `S` is a string of length 20. It is an e-mail message sent by Benjamin to Ali.
  - The function `SendA` should return a string. It is a latter written by Ali to Benjamin.
  - The return value should be a string whose length is between 1 and 300 000, inclusive. If this condition is not satisfied, your program is judged as **Wrong Answer [5]**.
  - Each character of the return value should be either `0` or `1`. If this condition is not satisfied, your program is judged as **Wrong Answer [6]**.

For each function call to `Init`, the following function should be called once for each airport. In total, it should be called N times.

- `void SetID(int p, int value)`
  - The parameter `p` means Ali sets the ID code for the airport `p`. Here $0 \leq$ `p` $\leq N-1$ should be satisfied. If this condition is not satisfied, your program is judged as **Wrong Answer [1]**.
  - The parameter `value` is the ID code for the airport specified by Ali. Here $0 \leq$ `value` $\leq 2N+19$ should be satisfied. If this condition is not satisfied, your program is judged as **Wrong Answer [2]**.
  - It is not allowed to call the function `SetID` with same parameter `p` more than once. If this condition is not satisfied, your program is judged as **Wrong Answer [3]**.
  - When the function `Init` terminates, the number of function calls to `SetID` should be N. If this condition is not satisfied, your program is judged as **Wrong Answer [4]**.

When the function call to `SetID` is judged as Wrong Answer, your program is terminated immediately.

The second file is `Benjamin.cpp`. It should implement Benjamin's strategy. It should implement the following two functions. The program should include `Benjamin.h` using the preprocessing directive `#include`.

- `std::string SendB(int N, int X, int Y)`
  This function implements Benjamin's strategy to send an e-mail message to Ali. For each scenario (see Grading for details), this function is called exactly once after the function `Init` is called.
  - The parameter `N` is the number of airports in Republic of JOI.
  - The parameter `X` is the ID code of the airport where the amusement park is located.
  - The parameter `Y` is the ID code of the airport where the hot spring is located.
  - The function `SendB` should return a string which is the e-mail message sent by Benjamin to Ali.
  - If the return value is not a string of length 20, your program is judged as **Wrong Answer [7]**.
  - Each character of the return value should be either `0` or `1`. If this condition is not satisfied, your program is judged as **Wrong Answer [8]**.
- `int Answer(std::string T)`
  This function should calculate the minimum number of airline routes Benjamin has to take to travel from the airport x to the airport y. For each scenario (see Grading for details), this function is called exactly once after the function `SendA` is called.
  - The parameter `T` is a string whose length is between 1 and 300 000, inclusive. It is the letter sent by Ali to Benjamin.
  - This function should return the minimum number of airline routes Benjamin has to take to travel from the airport x to the airport y.

Important Notices

- Your program can implement other functions for internal use, or use global variables. Submitted files will be compiled with the grader, and become a single executable file. All global variables and internal functions should be declared in an unnamed namespace to avoid confliction with other files. When it is graded, it will be executed as two processes of Ali and Benjamin. The process of Ali and the process of Benjamin cannot share global variables.
- Your program must not use the standard input and the standard output. Your program must not communicate with other files by any methods. However, your program may output debugging information to the standard error.

Grading

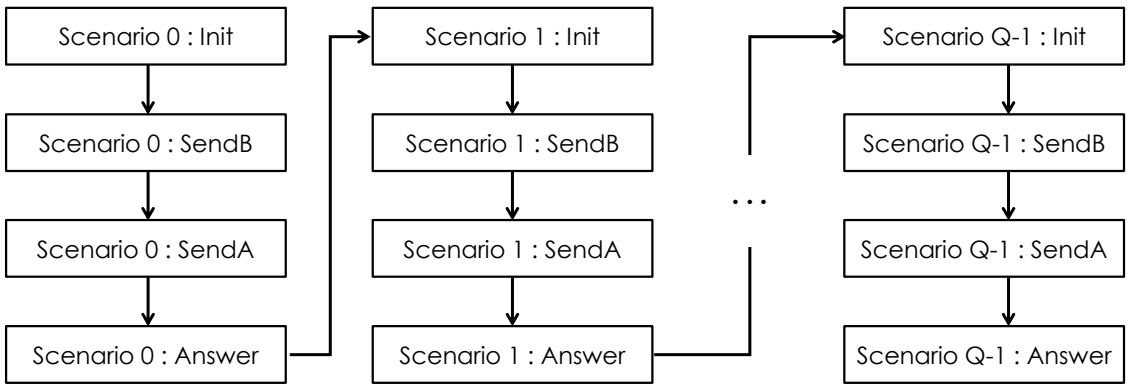
A test case consists of Q scenarios, numbered from 0 to $Q-1$. The following values are specified for each scenario. For the range of these values, see **Constraints**.

- The number N of airports in Republic of JOI.
- The airport number x of the airport where the amusement park is located.
- The airport number y of the airport where the hot spring is located.
- The information of the airline routes $(U_0, V_0), (U_1, V_1), \cdots, (U_{N-2}, V_{N-2})$.

For each scenario, the functions `Init`, `SendB`, `SendA`, `Answer` are called. Your program should call appropriate functions with valid parameters, and return appropriate values. These functions are called in the following order.

1. For $k = 0, 1, \cdots, Q-1$, the following procedures 2–5 are performed in this order.
2. The function `Init` is called. The parameters are set for the scenario k as specified in Implementation Details.
3. The function `SendB` is called. The parameters are set for the scenario k as specified in Implementation Details.
4. The function `SendA` is called. The parameters are set for the scenario k as specified in Implementation Details.
5. The function `Answer` is called. The parameters are set for the scenario k as specified in Implementation Details.

If your program is judged as Wrong Answer during these procedures, your program is terminated immediately, and your program is considered to fail the test case.



Compilation and Test Run

You can download an archive file from the contest webpage which contains the sample grader to test your program. The archive file also contains a sample source file of your program.

The sample grader is the file `grader.cpp`. In order to test your program, put `grader.cpp`, `Ali.cpp`, `Benjamin.cpp`, `Ali.h`, `Benjamin.h` in the same directory, and run the following command to compile your programs.

```
g++ -std=gnu++17 -O2 -o grader grader.cpp Ali.cpp Benjamin.cpp
```

When the compilation succeeds, the executable file `grader` is generated.

Note that the actual grader is different from the sample grader. The sample grader will be executed as a single process, which will read input data from the standard input and write the results to the standard output.

Input for the Sample Grader

The sample grader reads the following data from the standard input. The input values should be all integers.

Q
(Input for Scenario 0)
(Input for Scenario 1)
$\vdots$
(Input for Scenario $Q-1$)

The format of the input data for each scenario is as follows.

$N\ x\ y$
$U_0\ V_0$
$U_1\ V_1$
$\vdots$
$U_{N-2}\ V_{N-2}$

Output of the Sample Grader

If your program is judged as any one of Wrong Answer [1]-[8], the sample grader writes its type as "`Wrong Answer [1]`" (quotes for clarity).

Otherwise, for each scenario, it writes the return value of the function `Answer` and the maximum length of the strings sent by Ali to Benjamin. The sample grader does not check whether the return value of the function `Answer` is correct or not.

```
Scenario 0: Your Answer = 3
Scenario 1: Your Answer = 1
Scenario 2: Your Answer = 4
Scenario 3: Your Answer = 1
Scenario 4: Your Answer = 5
Accepted: Maximum Length = 24
```

If your program satisfies the conditions of several types of Wrong Answer, the sample grader reports only one of them. Moreover, if your program is not judged as Wrong Answer for the first scenario, the sample grader may output intermediate results as follows even if your program is judged as one of Wrong Answer [1]-[8] for a later scenario.

```
Scenario 0: Your Answer = 3
Scenario 1: Your Answer = 1
Scenario 2: Your Answer = 4
Wrong Answer [8]
```

Constraints

- $1 \leq Q \leq 50$.
- $2 \leq N \leq 10\,000$.
- $0 \leq U_i < V_i \leq N - 1$ $(0 \leq i \leq N - 2)$.
- $0 \leq x \leq N - 1$.
- $0 \leq y \leq N - 1$.
- $x \neq y$.
- It is possible to travel from any airport to any other airport by connecting several airline routes.
- For every airport, there are at most 3 airline routes connecting it with another airport.

Subtasks

1. (15 points) $Q = 1$.
2. (85 points) $Q \geq 2$.

Grading for Subtask 1

If your program is judged as Wrong Answer in any of the scenarios for Subtask 1, your score for this subtask is 0.

If your program answers all the test cases for Subtask 1 correctly, your score for this subtask is calculated as follows. Here, L_1 is the maximum length of the strings sent by Ali to Benjamin.

The value of L_1	Score
$150\,001 \leq L_1 \leq 300\,000$	7 points
$20\,001 \leq L_1 \leq 150\,000$	11 points
$L_1 \leq 20\,000$	15 points

Grading for Subtask 2

If your program is judged as Wrong Answer in any of the scenarios for Subtask 2, your score for this subtask is 0.

If your program answers all the test cases for Subtask 2 correctly, your score for this subtask is calculated as follows. Here, L_2 is the maximum length of the strings sent by Ali to Benjamin. **Note that if** $1\,401 \leq L_2$**, your score for this subtask is** 0**.**

The value of L_2	Score
$1\,401 \leq L_2 \leq 300\,000$	0 points
$71 \leq L_2 \leq 1\,400$	$52 - 35 \times \log_{10}\left(\frac{L_2}{70}\right)$ points (rounded down to the nearest integer)
$45 \leq L_2 \leq 70$	$87 - 0.5 \times L_2$ points (rounded down to the nearest integer)
$25 \leq L_2 \leq 44$	$109 - L_2$ points
$L_2 \leq 24$	85 points

Sample Communication

Here is a sample input for the sample grader and corresponding function calls. In the following example, the ID codes of the airports $0, 1, 2, 3$ set by Ali by calling the function `Init` are $12, 21, 25, 27$, respectively.

Sample Input 1
1 4 0 2 0 1 1 2 2 3

Ali's Call	Ali's Return Value	Benjamin's Call	Benjamin's Return Value
`Init(4,[0,1,2],[1,2,3])`			
`SetID(0,12)`			
`SetID(1,21)`			
`SetID(2,25)`			
`SetID(3,27)`			
		`SendB(12,25)`	`"00000111110000011111"`
`SendA("00...11")`	`"10"`		
		`Answer("10")`	`2`

In this sample input, there are N (= 4) airports and three airline routes.

- An airline route connecting the airport 0 with the airport 1.
- An airline route connecting the airport 1 with the airport 2.
- An airline route connecting the airport 2 with the airport 3.

Since we need to take at least two airilne routes to travel from the airport x (= 0) to the airport y (= 2), the function `Answer` should return `2`.

Note that the return value of the function `SendB` is not the airport numbers $(x, y) = (0, 2)$. It returns the ID codes of the airports $(X, Y) = (12, 25)$.

Sample Input 2

```
2
10 0 9
0 1
1 2
2 3
3 4
4 5
5 6
6 7
7 8
8 9
15 12 8
0 1
0 2
1 3
1 4
2 5
2 6
3 7
3 8
4 9
4 10
5 11
5 12
6 13
6 14
```

In this sample input, there are $Q = 2$ scenarios.

- For the first scenario, the function `Answer` should return `9`.
- For the second scenario, the function `Answer` should return `6`.