Building: ビルの飾りつけ

input ファイル “building.in”
output 標準出力
ソースファイル building.c/building.cpp/Building.java
時間制限 1 秒 / データ

国際情報オリンピックが日本で開かれることとなり，世界の選手達を歓迎するため，空港から宿泊施設までの道沿いにある高層ビルを飾りつけることにした．ある著名なデザイナーにデザインを依頼したところ，飾りつけに利用するビルは，空港から宿泊施設に向けて高くなっていく必要があると言った．つまり，飾りつけに利用するビルの高さを，空港に近いものから順に $h_1, h_2, h_3, \cdots$ とおくと，$h_1 < h_2 < h_3 < \cdots$ となっていなければならない．

できるだけ飾りつけを華やかにするため，飾りつけに利用するビルの数をできるだけ多くしたい．入力として全てのビルの高さが与えられたとき，利用することのできるビルの数の最大値を計算するプログラムを作れ．

Input. 入力ファイル building.in の 1 行目には，1 つの整数 n $(1 \leqq n \leqq 1000)$ が書かれている．これは，空港から宿泊施設までの道のりにあるビルの数が n であることを表す．

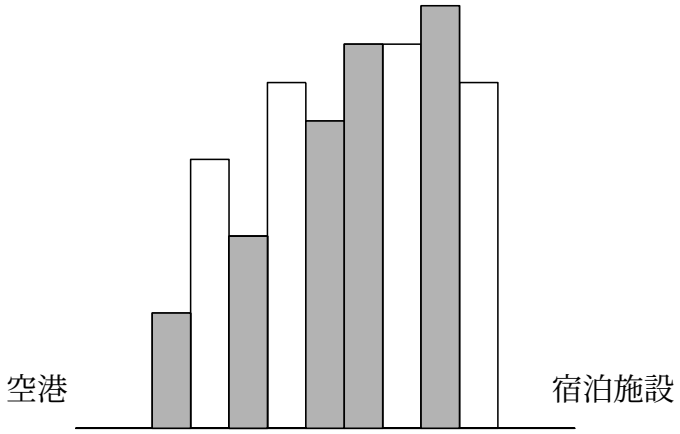
次の n 行には，それぞれのビルの高さが書かれてる．$i+1$ 行目 $(1 \leqq i \leqq n)$ には 1 つの整数 $a_i (1 \leqq a_i \leqq 10000)$ が書かれている．これは，空港から i 個目のビルの高さが a_i であることを表す．

Output. 出力は，標準出力に行うこと．飾りつけに利用することのできるビルの数の最大値を表す整数を出力せよ．

例

building.in	標準出力
9	5
3	
7	
5	
9	
8	
10	
10	
11	
9	

ビルの総数が 9 であり、高さが空港に近いものから順に 3, 7, 5, 9, 8, 10, 10, 11, 9 である場合を示したのが下図である。色をつけて示したビルを選ぶことにより、飾り付けに利用するビルの数を最大にすることができる。このときの最大値は 5 である。



Fermat: フェルマー方程式

input ファイル “fermat.in”
output 標準出力
ソースファイル fermat.c/fermat.cpp/Fermat.java
時間制限 0.5 秒 / データ

素数 p と自然数 $n \geqq 1$ が与えられたとき,

$$x^n + y^n \equiv z^n \pmod{p}$$

をみたす整数 x, y, z $(0 \leqq x, y, z \leqq p-1)$ の組 (x, y, z) の個数 m を求めるプログラムを作れ．ここで，$a \equiv b \pmod{p}$ とは，$a - b$ が p で割り切れることを意味する．

Input. 入力ファイル fermat.in の 1 行目には素数 p $(p < 10000)$ が書かれている．2 行目には自然数 n $(1 \leqq n \leqq 10000)$ が書かれている．

Output. 標準出力に 1 つの整数 m のみからなる 1 行を出力せよ．

例 1

fermat.in	標準出力
3	9
5	

$x^5 + y^5 \equiv z^5 \pmod{3}$ をみたす整数 x, y, z $(0 \leqq x, y, z \leqq 2)$ の組 (x, y, z) は

(0,0,0), (0,1,1), (0,2,2), (1,0,1), (1,1,2), (1,2,0), (2,0,2), (2,1,0), (2,2,1)

の 9 個である．

例 2

fermat.in	標準出力
19	487
21	

$x^{21} + y^{21} \equiv z^{21} \pmod{19}$ をみたす整数 x, y, z $(0 \leqq x, y, z \leqq 18)$ の組 (x, y, z) は 487 個ある．

注意 採点に用いる入力データに対しては，m の値は 2^{31} よりも小さい．

Salt: SALT TREE XV

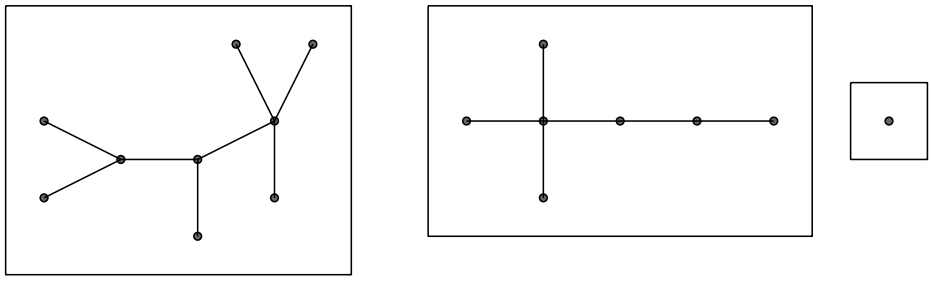
input ファイル “salt.in”
output なし
ソースファイル salt.c/salt.cpp/Salt.java
時間制限 3 秒 / データ

SALT TREE XV (“salt tree the fifteenth” と読む) は二人のプレイヤーで遊ぶゲームである。最初二人には一つの木が与えられる。二人は交互に木の辺か頂点を取る。プレイヤーが辺を取った場合、単にその辺が消滅する。プレイヤーが頂点を取った場合、その頂点だけでなく頂点に接続している辺があればそれらの辺も消滅する。最後の頂点 (LAST VERTEX) を取ったプレイヤーが勝者となる。

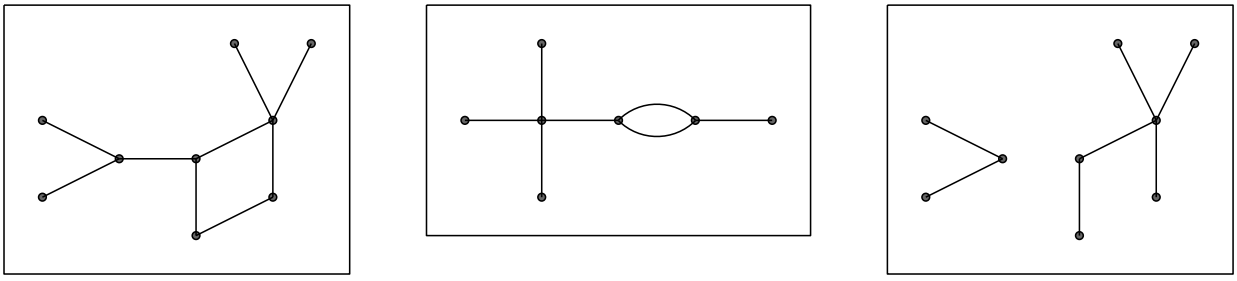
SALT TREE XV で先手が最善を尽くした場合、後手がどんな手を打っても必ず先手が勝つことが証明できる。先手としてプレイして必ず勝つプログラムを書け。

補足

頂点を一つ以上持ち、閉路を持たない連結なグラフを木という。例えば、次の三つはどれも木である。

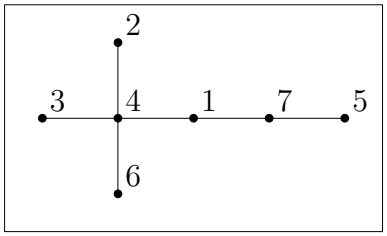


次の三つのグラフはどれも木でない。左下のグラフと中央のグラフは閉路を持つので木ではない。右下のグラフは連結でないので木ではない。

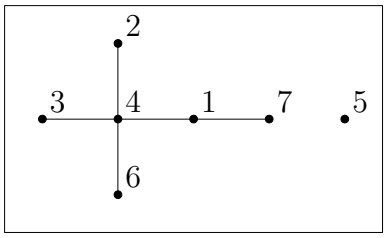


例

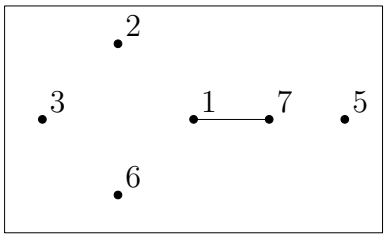
次の木が与えられた場合を考える。



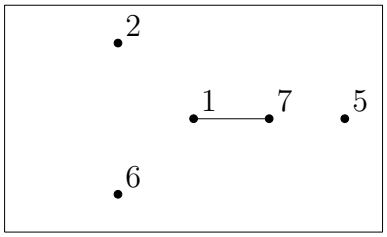
先手が辺 $(5, 7)$ を取ると、次のようになる。



続いて後手が頂点 4 を取ると、次のようになる。



続いて先手が頂点 3 を取ると、次のようになる。



入力

プログラムは salt.in から木のデータを読み込む。 salt.in は次のような構造をしている。この例は上の木に対応している。

```
7
3 4
5 7
4 6
2 4
1 4
1 7
```

1 行目には木の頂点の個数 n $(1 \leqq n \leqq 1000)$ が書かれている。木の頂点は 1 から n までの整数でラベル付けされる。 2 行目から n 行目までの $n-1$ 行には木の辺の両端の頂点の番号が書かれている。各辺はちょうど 1 度ずつ出現し、出現順には意味がない。各辺はラベルの小さい頂点、大きい頂点の順に書かれる。

インターフェース

プログラムは以下のインターフェースを通して後手のプログラムと通信する。

C/C++ 版

次のようにしてインクルードファイル saltc.h をインクルードすること。

```
#include "saltc.h"
```

プログラムはライブラリーの次の関数を呼ぶ必要がある。

`void turn(int u, int v, int* ru, int* rv);` 最初の二つのパラメーターは先手 (あなた) の手番に行うことを指定する。u = v のときは、頂点 u を取ることを意味する。u < v のときは、頂点 u と頂点 v を結ぶ辺を取ることを意味する。u > v であってはいけない。後の二つのパラメーターでは、後手の手番に行われたことを読み取る。ru と rv は有効なポインターでなければならず、また同じ領域を指していてはいけない。*ru と *rv は上の u, v と同じ意味を持つ。 *ru > *rv となることはない。ただし、あなたが最後の頂点を取った場合は、 *ru と *rv は 0 となる。この場合は、他の API 関数を呼ばずに終了しなければならない。

saltc.o のソースファイルは saltc.c という名前で置いてある。
プログラムは、 saltc.o と saltc.h をカレントディレクトリにコピーして、

```
gcc -O2 -lm salt.c saltc.o -o salt    (C 言語の場合)
g++ -O2 -lm salt.cpp saltc.o -o salt  (C++ の場合)
```

とすることでコンパイルできる。

Java 版

プログラムはライブラリーの次の関数を呼ぶ必要がある。

`static int[] Saltj.turn(int u, int v);` パラメーターは先手 (あなた) の手番に行うことを指定する。u = v のときは、頂点 u を取ることを意味する。u < v のときは、頂点 u と頂点 v を結ぶ辺を取ることを意味する。u > v であってはいけない。返り値は int 型のサイズ 2 の配列である。この配列を r とする。r[0] と r[1] は、後手の手番に行われたことを表す。r[0] と r[1] は上の u, v と同じ意味を持つ。 r[0] > r[1] となることはない。ただし、あなたが最後の頂点を取った場合は、 r[0] と r[1] は 0 となる。この場合は、他の API 関数を呼ばずに終了しなければならない。

Saltj.class のソースファイルは Saltj.java という名前で置いてある。
プログラムをコンパイルするときは、 Saltj.class をカレントディレクトリにコピーしておく必要がある。

出力

プログラムは何も出力してはいけない。

動作確認用プログラム

実行ファイルを作ったら、動作確認のため、ランダムな手を返す後手のプログラムと対戦させることができる。salt-fake という名前で入っている。なお、このプログラムは頂点が 15 個以下のグラフでしか動作しないようになっている。
カレントディレクトリに salt.in を置き、

```
./salt-fake ./salt          (C/C++ の場合)
./salt-fake java Salt       (Java の場合)
```

のように呼び出す。

先手プログラムと後手プログラムは標準入出力を通して通信をするので、標準入出力を使うと予測のつかない結果になる可能性があることに注意せよ。

`salt-fake` はあくまでも受験者が動作確認のために使用できるように提供されるだけであり、実際の採点ではこれとは異なる方法で実行する可能性がある。

採点

テストデータは複数用意されている。各テストデータにつき、何度かゲームを行い、プログラムがつねに勝てばそのテストデータの分の点数を与える。プログラムがゲームのルールに照らして正しくない手を指定した場合や、ゲームに負けた場合、ゲームに勝つ前に終了した場合は、そのテストデータの点数は与えない。