Zadanie «Ochrona» (och)

Muzeum Historii Czwartorzędu jest wielkim budynkiem z wieloma korytarzami. Korytarz jest prostym odcinkiem i łączy się z innymi korytarzami skrzyżowaniem. W skrzyżowaniu może łączyć się wiele korytarzy. Między dowolnymi dwoma miejscami w muzeum istnieje dokładnie jedna droga. Budynku strzegą strażnicy. Każdy z nich stoi na skrzyżowaniu korytarzy i widzi, co się dzieje w korytarzu aż do następnego skrzyżowania. Strażnik nie widzi, co się dzieje za skrzyżowaniem. Inaczej mówiąc strażnik pilnuje korytarzy, które rozchodzą się od skrzyżowania, na którym stoi, aż do końca korytarza, czyli następnego skrzyżowania. Jest możliwe, że tego samego korytarza strzeże dwóch strażników stojących na kończących korytarz skrzyżowaniach. Firma ochroniarska chce wiedzieć, ilu strażników jest potrzebnych do strzeżenia budynku.

Proszę napisać program, który będzie znajdował minimalną liczbę strażników konieczną do strzeżenia budynku.

Specyfikacja wejścia

Dane wejściowe zawarte są w pliku tekstowym. W pliku wejściowym najpierw podana jest liczba korytarzy w budynku (jedna liczba w linii). Następnie podane są korytarze. Każdy korytarz budynku jest zapisany w jednej linii pliku w postaci dwóch liczb, które są numerami skrzyżowań budynku.

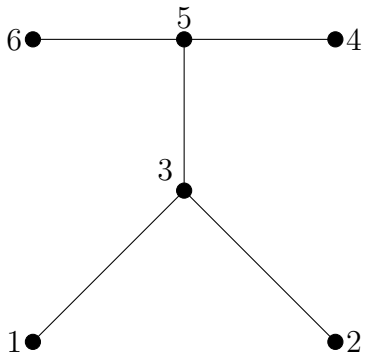
Specyfikacja wyjścia

W pliku wynikowym należy podać wyłącznie jedną liczbę: minimalną liczbę strażników dla budynku.

Przykładowe wejście

```
5

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



Przykładowe wyjście

Wystarczy, że strażnicy będą stali na krzyżowaniach o numerach: 3 i 5, zatem wystarczy tylko dwóch strażników. Plik wynikowy będzie miał postać:

```
2
```

Ustalenia techniczne

1. Rozwiązaniem zadania są:

   - program konsolowy napisany w języku C/C++ – Kod źródłowy programu powinien być zawarty wyłącznie w jednym pliku o nazwie `och.c` (dla języka C) lub `och.cpp` (dla języka C++). W pierwszej linii pliku źródłowego należy umieścić w komentarzu *indywidualny kod uczestnika* (IKU). Nie jest dopuszczalne umieszczanie w kodzie jakichkolwiek innych danych umożliwiających zidentyfikowanie uczestnika (także we właściwościach pliku).
   - pliki wyjściowe wypracowane przez program dla danych testowych dostarczonych wraz z treścią zadania – Pliki muszą być nazwane zgodnie z niżej umieszczonym nazewnictwem. Pliki muszą być zgodne ze specyfikacją wyjścia.

   Wszystkie powyższe pliki należy spakować do pliku `IKU-och.zip`, gdzie `IKU` jest *indywidualnym kodem uczestnika*.

2. Program powinien odczytywać dane wejściowe z pliku o nazwie podanej w treści zadania, a wynik należy zapisać też do pliku, którego nazwa jest podana w treści zadania.

3. Należy przyjąć, że dane wejściowe mają poprawny format (opisany w treści zadania). Plik wyjściowy powinien mieć format opisany w treści zadania.

4. W programach można korzystać wyłącznie ze standardowych bibliotek języka C/C++.

5. W programach nie można korzystać z rozwiązań i mechanizmów nieprzenośnych (np. zależnych od systemu operacyjnego).

6. Programy nie mogą:

   - tworzyć nowych procesów lub wątków,
   - uruchamiać innych programów,
   - używać funkcji sieciowych (np. `socket`, `send` itp.),

- oczekiwać na interakcję użytkownika.

7. Zadanie należy przesłać przez stronę konkursu «Złoty Indeks» Platformy Zdalnej Edukacji `https://platforma.polsl.pl/rd/course/view.php?id=7` korzystając z łącza do przesyłania rozwiązań zadania «och».

8. Zadanie jest oceniane w skali 0-15 punktów.

Nazewnictwo plików

Należy ściśle przestrzegać nazewnictwa plików wynikowych. Każdemu plikowi wejściowemu odpowiada odpowiada jeden plik wyjściowy:

plik wejściowy	plik wyjściowy
`och-01.in`	`och-01.out`
`och-02.in`	`och-02.out`
`och-03.in`	`och-03.out`
`och-04.in`	`och-04.out`
`och-05.in`	`och-05.out`
`och-06.in`	`och-06.out`
`och-07.in`	`och-07.out`
`och-08.in`	`och-08.out`
`och-09.in`	`och-09.out`
`och-10.in`	`och-10.out`
`och-11.in`	`och-11.out`
`och-12.in`	`och-12.out`
`och-13.in`	`och-13.out`
`och-14.in`	`och-14.out`
`och-15.in`	`och-15.out`