Leludisain

Ülesande nimi	ToyDesign
Sisend	Interaktiivne ülesanne
Väljund	Interaktiivne ülesanne
Ajapiirang	1 sekund
Mälupiirang	256 megabaiti

Töötad firma heaks, mis loob lelusid. Uue lelu loomine käib nii: kastist ulatuvad välja n viiku, nummerdatud 1 kuni n. Mõned viikude paarid on kasti sees ühendatud juhtmetega. (Teiste sõnadega moodustavad viigud ja juhtmed suunamata graafi, kus viigud on tipud ja juhtmed on servad.) Juhtmeid väljastpoolt ei näe ja ainus viis nende kohta midagi teada saada on **testrit** kasutades: võime valida kaks viiku i ja j, kus $i \neq j$, ja tester ütleb, kas need kaks viiku on kasti sees omavahel ühendatud, juhtmega või kaudselt. (Seega ütleb tester, kas nende viikude vahel on graafis servadeahel.)

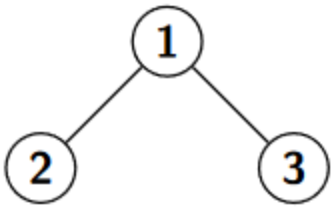
Kutsume kasti sees olevat juhtmekomplekti lelu **disainiks**.

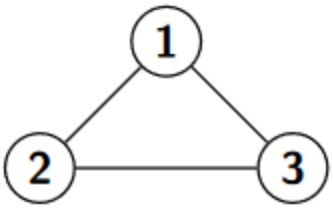
Kasutad disainide loomiseks ja nende kohta pärimiseks erilist tarkvara. Tarkvara töötab nii: see alustab mingi leludisainiga, mille nimeks on "disain 0". See ei näita selle disaini juhtmeid kasti sees. Selle asemel võid korduvalt rakendada järgmist kolmest sammust koosnevat operatsiooni:

1. Vali disaini number a ja kahe viigu numbrid i ja j, nii et $i \neq j$.
2. Tarkvara ütleb sulle, mis juhtuks, kui me kasutaksime testrit nende kahe viigu peal. Teiste sõnadega ütleb see, kas viigud i ja j on disainis a omavahel ühenduses (otse või kaudselt).
3. Kui disainis a ei olnud valitud viigud otse või kaudselt ühenduses, siis loob tarkvara uue disaini, milles on kõik juhtmed disainist a ja lisaks veel üks juhe viikude i ja j vahel. Sellele disainile antakse järgmine saadaolev järjekorranumber. (See tähendab, et esimene nii loodud disain saab järjekorranumbri 1, järgmine 2 jne.) Pane tähele, et see ei muuda disaini a, vaid loob lihtsalt uue disaini, milles on üks lisajuhe.

Sinu ülesanne on seda operatsiooni kasutades saada võimalikult palju teada disain 0 kohta.

Pane tähele, et alati ei ole võimalik leida täpset juhtmekomplekti disain 0 jaoks, kuna puudub viis eristada juhtmete otseseid ja kaudseid ühendusi. Vaatame näiteks järgmiseid kaht disaini, kus $n = 3$:





Tester annaks mõlemas disainis kõigi viigupaaride puhul teada, et need on ühendatud, seega ei ole meil võimalik neid disaine ülalkirjeldatud tarkvaraga eristada.

Sinu ülesanne on leida ükskõik milline disain, mis on samaväärne disainiga 0. Kaks disaini on **samaväärsed**, kui testija annab mõlemas disainis kõigi viigupaaride jaoks sama tulemuse.

Implementatsioon

See on interaktiivne ülesanne. Pead kirjutama funktsiooni

```
void ToyDesign(int n, int max_ops);
```

mis leiab disain 0-ga *samaväärse* disaini. Selleks saab su lahendus kutsuda välja kaht järgmist funktsiooni. Esimene neist on:

```
int Connected(int a, int i, int j);
```

kus $1 \leq i, j \leq n, i \neq j, a \geq 0$ ja a ei tohi olla suurem kui seni loodud disainide arv. Kui viigud i ja j on disainis a (otse või kaudselt) ühendatud, siis tagastab see a. Vastasel juhul tagastab see seni loodud disainide arvu pluss üks, mis saab olema uue disaini number (disain, kus on kõik juhtmed disainist a ja lisaks juhe i ja j vahel). Funktsiooni `Connected` saab välja kutsuda ülimalt `max_ops` korda.

Kui su programm on lõpetanud kõik oma `Connected` väljakutsed, siis peaks ta kirjeldama disaini, mis on samaväärne disainiga 0. Disaini kirjeldamiseks peaks programm välja kutsuma

```
void DescribeDesign(std::vector<std::pair<int,int>> result);
```

Parameeter `result` on täisarvupaaride vektor, mis kirjeldab viikude vahelisi juhtmeid. Iga paar vastab ühele juhtmele ja peab sisaldama mõlema juhtmega ühendatud viigu numbreid. Iga järjestamata paari viikude vahel võib olla maksimaalselt üks juhe. Juhtme mõlemad otsad ei tohi olla ühendatud ühe ja sama viiguga. Sellise funktsiooni väljakutsumine lõpetab sinu programmi töö.

Piirangud

- $2 \leq n \leq 200$

Alamülesanded

1. (10 punkti): $n \leq 200$, *max_ops* $= 20\,000$
2. (20 punkti): $n \leq 8$, *max_ops* $= 20$
3. (35 punkti): $n \leq 200$, *max_ops* $= 2\,000$
4. (35 punkti): $n \leq 200$, *max_ops* $= 1\,350$

Näidissuhtlus

Võistleja tegevus	Hindaja tegevus	Selgitus
	`ToyDesign(4, 20)`	Lelul on 4 viiku. Pead leidma ükskõik millise disain 0-ga võrdväärse disaini, kutsudes välja funktsiooni `Connected` ülimalt 20 korda.
`Connected(0, 1, 2)`	`Tagastab 1.`	Viigud 1 ja 2 ei ole disainis 0 otse ega kaudselt ühendatud. Luuakse uus disain numbriga 1.
`Connected(1, 3, 2)`	`Tagastab 2.`	Viigud 3 ja 2 ei ole otse ega kaudselt disainis 1 ühendatud. Luuakse uus disain numbriga 2.
`Connected(0, 3, 4)`	`Tagastab 0.`	Viigud 3 ja 4 on disainis 0 otseselt või kaudselt ühendatud. Uut disaini ei looda.
`DescribeDesign({{3, 4}})`	-	Kirjeldame disaini, milles on vaid üks juhe: viikude 3 ja 4 vahel.

Näidishindaja

Ülesande manuses `ToyDesign.zip` antud näidishindaja `grader.cpp` loeb sisendit standardsisendist järgnevas vormingus:

- Esimesel real on viikude arv n, juhtmete arv m ja *max_ops*.
- Järgmisel m real on viigupaaridena kirjeldatud juhtmed.

Näidishindaja loeb sisendit ja kutsub välja sinu lahenduses oleva funktsiooni `ToyDesign`. Hindaja väljastab seejärel sinu programmi käitumise põhjal ühe järgmistest teavitustest:

- "Wrong answer: Number of operations exceeds the limit.", kui funktsiooni `Connected` väljakutsete arv ületab *max_ops*
- "Wrong answer: Wrong design id.", kui argumendiga a viidatud disain funktsiooni `Connected` kutsumisel (veel) ei eksisteeri.
- "Wrong answer: Incorrect design.", kui funktsiooni `DescribeDesign` abil kirjeldatud disain ei ole samaväärne disainiga 0.
- "OK!" kui funtsiooni `DescribeDesign` abil kirjeldatud disain on samaväärne disainiga 0.

Näidishindaja oma lahendusega kompileerimiseks võid terminalis kasutada järgmist käsku:

```
g++ -std=gnu++11 -O2 -o solution grader.cpp solution.cpp
```

kus `solution.cpp` on sinu lahendusfail, mille kavatsed esitada CMSis. Oma programmi käivitamiseks manuses toodud näidissisendiga sisesta terminali järgmine käsk:

```
./solution < input.txt
```