KANTONALNO TAKMIČENJE
IZ INFORMATIKE ZA UČENIKE SREDNJIH ŠKOLA 2015

Sarajevo, april 2015. godine

KANTONALNO TAKMIČENJE IZ INFORMATIKE
za učenike srednjih škola sa područja Kantona Sarajevo

SPISAK PRIJAVLJENIH ŠKOLA

Redni broj	Škola	R/B	Takmičari	Programski jezik	Napomena
1.	Gimnazija „Dobrinja“	1.	1. Teskeredžić Armin	C++	
		2.	2. Rahić Ćazim	C++	
		3.	3. Turčalo Benjamin	C++	
2.	SŠC „Nedžad Ibrišimović“	4.	1. Muratović Nedim	C++	
		5.	2. Fazlić Faris	C++	
3.	Treća gimnazija Sarajevo	6.	1. Arnautović Adnan	C++	
		7.	2. Aćimović Dejan	C++	
		8.	3. Hadžirović Kenan	C++	
4.	Druga gimnazija Sarajevo	9.	1. Muamer Parić	C++	
		10.	2. Ali Husić	C++	
		11.	3. Adnan Brđanin	C++	
		12.	4. Vedad Fejzagić	C++	
		13.	5. Kerim Jamaković	C++	
		14.	6. Erol Balković	C++	
5.	Srednjoškolski centar Hadžići	15.	1. Šarić Irhad	C++	
		16.	2. Ćosić Rijad	C++	
6.	Srednja elektrotehnička škola za energetiku	17.	1. Murtić Alma	T Pascal 1.5	
		18.	2. Stanić Ivica	T Pascal 1.5	
		19.	3. Pohara Amar	T Pascal 1.5	
		20.	4. Rizvo Enver	T Pascal 1.5	
		21.	5. Sirbubalo Eldar	T Pascal 1.5	
7.	Srednjoškolski centar Vogošća	22.	1. Hakimi Ilijad	C++	
		23.	2. Softić Alen	C++	
		24.	3. Šetkić Mirsad	C++	

KANTONALNO TAKMIČENJE IZ INFORMATIKE
za učenike srednjih škola sa područja Kantona Sarajevo

SPISAK PRIJAVLJENIH ŠKOLA

Redni broj	Škola	R/B	Takmičari	Programski jezik	Napomena
8.	Srednja elektrotehnička škola	25.	1. Paško Zdilar	C++	
		26.	2. Almedin Mrnđić	C++	
		27.	3. Semir Suljević	C++	
9.	Perzijsko-bosanski koledž sa internatom	28.	1. Đino Ajdin	C++	
		29.	2. Feriz Emina	C++	
		30.	3. Suljić Akif	Pascal	
10.	Gimnazija Obala	31.	1. Šehalić Mirza	C++	
		32.	2. Krek Zejd	C++	
		33.	3. Vuk Alen	C++	
11.	Prva bošnjačka gimnazija	34.	1. Berilo Haris	C/C++	
		35.	2. Jajčanin Ajdin	C/C++	
		36.	3. Stanbolić Selvin Denis	C/C++	
12.	KŠC Opća-Realna Gimnazija	37.	1. Filip Vlaisavljević	C++	
		38.	2. David Dragan Hadžić	C++	
		39.	3. Patrik Šajinović	C++	
13.	Srednja Međunarodna škola	40.	1. Smajević Irfan	C++	
		41.	2. Alibašić Faruk	C++	
		42.	3. Ašimović Armin	C++	
		43.	4. Crnčević Elvir	C++	
		44.	5. Banjac Miron	C++	
		45.	6. Hanjalić Haris	C++	
14.	Četvrta gimnazija	46.	1. Bilal Armin	C++	
		47.	2. Kerim Redžepagić	C++	
		48.	3. Adna Tutin	C++	

KANTONALNO TAKMIČENJE IZ INFORMATIKE
za učenike srednjih škola sa područja Kantona Sarajevo

SPISAK PRIJAVLJENIH ŠKOLA

Redni broj	Škola	R/B	Takmičari	Programski jezik	Napomena
15.	**Prva Gimnazija**	**49.**	1. Memišević Nedim	C++	
		50.	2. Jamaković Nihad	C++	
		51.	3. Šoše Adi	C++	
16.	**Sarajevo Collage**	**52.**	1. Fatih Zukorlić	C++	

PROGRAM TAKMIČENJA

08:15-08:30	Okupljanje ekipa
08:30-09:00	Otvaranje takmičenja, prezentacija domaćina i izvlačenje takmičarskih šifri
09:00-12:00	Pojedinačno takmičenje
12:00-12:45	Pregled zadataka i priprema preliminarnih rezultata
12:15-12:45	Ručak
12:45-13:30	Uvid u zadatke i eventualne reklamacije
13:45-14:15	**Proglašenje pobjednika**

Konačni rezultati Kantonalnog takmičenja iz Informatike za učenike srednjih škola

Mjesto	Ime	Prezime	Škola	I	II	III	IV	Ukupno
1	**Armin**	**Ašimović**	**Srednja međunarodna škola**	40	100	40	70	**250**
2	**Ali**	**Husić**	**Druga gimnazija Sarajevo**	0	100	30	100	**230**
3	**Faruk**	**Alibašić**	**Srednja međunarodna škola**	0	100	30	90	**220**
4	Elvir	Crnčević	Srednja međunarodna škola	20	60	30	100	**210**
5	Paško	Zdilar	Srednja elektrotehnička škola	0	100	30	70	**200**
6	Vedad	Fejzagić	Druga gimnazija Sarajevo	0	100	0	90	**190**
6	Erol	Balković	Druga gimnazija Sarajevo	0	100	0	90	**190**
8	Muamer	Parić	Druga gimnazija Sarajevo	0	100	0	70	**170**
8	Fatih	Zukorlić	Sarajevo koledž	0	100	0	70	**170**
10	Mirza	Šehalić	Gimnazija Obala	0	20	0	90	**110**
10	Zejd	Krek	Gimnazija Obala	0	20	0	90	**110**
10	Almedin	Mrnđić	Srednja elektrotehnička škola	0	100	0	10	**110**
13	Nihad	Jamaković	Prva gimnazija	0	100	0	0	**100**
13	Nedim	Memišević	Prva gimnazija	0	100	0	0	**100**
13	Kerim	Jamaković	Druga gimnazija Sarajevo	0	100	0	0	**100**
13	Adnan	Brđanin	Druga gimnazija Sarajevo	0	0	30	70	**100**
13	Adnan	Arnautović	Treća gimnazija Sarajevo	0	100	0	0	**100**
13	Armin	Bilal	Četvrta gimnazija	0	100	0	0	**100**
19	Dejan	Aćimović	Treća gimnazija Sarajevo	0	30	0	0	**30**
20	Armin	Teskeredžić	Gimnazija Dobrinja	0	0	0	20	**20**
21	Ćazim	Rahić	Gimnazija Dobrinja	0	0	0	0	**0**
21	Amar	Pohara	SETŠ za energetiku	0	0	0	0	**0**
21	Alma	Murtić	SETŠ za energetiku	0	0	0	0	**0**
21	Kenan	Hadžirović	Treća gimnazija Sarajevo	0	0	0	0	**0**
21	Faris	Fazlić	SŠC Nedžad Ibrišimović	0	0	0	0	**0**
21	Nedim	Muratović	SŠC Nedžad Ibrišimović	0	0	0	0	**0**
21	Ivica	Stanić	SETŠ za energetiku	0	0	0	0	**0**
21	Eldar	Sirbubalo	SETŠ za energetiku	0	0	0	0	**0**
21	Kerim	Redžepagić	Četvrta gimnazija	0	0	0	0	**0**
21	Enver	Rizvo	SETŠ za energetiku	0	0	0	0	**0**
21	Benjamin	Turčalo	Gimnazija Dobrinja	0	0	0	0	**0**
21	Riad	Ćosić	Srednjoškolski centar Hadžići	0	0	0	0	**0**
21	Irhad	Šarić	Srednjoškolski centar Hadžići	0	0	0	0	**0**
21	Haris	Deleut	Srednja međunarodna škola	0	0	0	0	**0**
21	Haris	Hanjalić	Srednja međunarodna škola	0	0	0	0	**0**
21	Adi	Šoše	Prva gimnazija	0	0	0	0	**0**
21	Selvin Denis	Stambolić	Prva bošnjačka gimnazija	0	0	0	0	**0**
21	Ajdin	Jajčanin	Prva bošnjačka gimnazija	0	0	0	0	**0**

Konačni rezultati Kantonalnog takmičenja iz Informatike za učenike srednjih škola

Mjesto	Ime	Prezime	Škola	I	II	III	IV	Ukupno
21	Haris	Berilo	Prva bošnjačka gimnazija	0	0	0	0	**0**
21	Ilijad	Hakimi	Srednjoškolski centar Vogošća	0	0	0	0	**0**
21	Alen	Softić	Srednjoškolski centar Vogošća	0	0	0	0	**0**
21	Semir	Suljević	Srednja elektrotehnička škola	0	0	0	0	**0**
21	Ajdin	Đino	Perzijsko – bosanski koledž	0	0	0	0	**0**
21	Adna	Tutun	Četvrta gimnazija	0	0	0	0	**0**
21	Filip	Vlaisavljević	KŠC Opća realna gimnazija	0	0	0	0	**0**
21	David Dragan	Hadžić	KŠC Opća realna gimnazija	0	0	0	0	**0**
21	Patrik	Šajinović	KŠC Opća realna gimnazija	0	0	0	0	**0**
21	Emina	Feriz	Perzijsko – bosanski koledž	0	0	0	0	**0**
21	Akif	Suljić	Perzijsko – bosanski koledž	0	0	0	0	**0**

Plasman škola

Mjesto	Škola	Takmičar 1	Takmičar 2	Ukupno
1	Srednja međunarodna škola	250	220	**470**
2	Druga gimnazija Sarajevo	230	190	**420**
3	Srednja elektrotehnička škola	200	110	**310**
4	Gimnazija Obala	110	110	**220**
5	Prva gimnazija	100	100	**200**
6	Sarajevo koledž	170	0	**170**
7	Treća gimnazija Sarajevo	100	30	**130**
8	Četvrta gimnazija	100	0	**100**
9	Gimnazija Dobrinja	20	0	**20**
10	SETŠ za energetiku	0	0	**0**
10	SŠC Nedžad Ibrišimović	0	0	**0**
10	Srednjoškolski centar Hadžići	0	0	**0**
10	Prva bošnjačka gimnazija	0	0	**0**
10	Perzijsko – bosanski koledž	0	0	**0**
10	KŠC Opća realna gimnazija	0	0	**0**
10	Srednjoškolski centar Vogošća	0	0	**0**

ZADACI

Takmičenje (takmicenje)

Najveći *hackaton* na svijetu upravo se bliži kraju! U proteklih 72 sata, hiljade programera širom svijeta se takmičilo u rješavanju niza interesantnih zadataka iz oblasti programiranja. Svaki takmičar je imao mogućnost da riješi jedan od oko 100 zadataka. Velike međunarodne korporacije koje su sponzorisale ovo takmičenje obezbijedile su *cluster* moćnih superračunara koji su neprekidno kompajlirali i testirali rješenja. Za svaki zadatak u prosjeku je dato oko 10 testova pri čemu je svaki uspješan test nosio po 1 bod, što znači da je svaki takmičar osvojio između 0 i 1000 bodova.

Pošto su dakle rezultati dostupni praktično istog trena, takmičarska komisija je odlučila da takmičarima odmah po završetku ponudi preliminarne rezultate putem weba, na način da takmičar na web stranici može ukucati svoju šifru i dobiti svoj rezultat u obliku:

```
Osvojili ste 175 bodova. Nalazite se na 387 mjestu od 1251 takmicara.
```

Softver koji prikazuje ove rezultate je spreman, a finalna top-lista takmičara bi bila gotova nekada sutra ili prekosutra.

No upravo sada, manje od 1h prije kraja, takmičarska komisija je otkrila jednu zabrinjavajuću činjenicu. Naime, kako su takmičari dobri programeri, napravili su skripte koje svakih 1 sekundu ili manje provjeravaju njihov rezultat, tako da u ovom trenutku web stranica takmičenja dobija na hiljade zahtjeva u sekundi! Ukoliko bi se sada objavili rezultati, web server bi se istog trena srušio.

Odlučili su da angažuju *tebe* da riješiš ovaj problem. Tvoj zadatak je da napraviš program koji na osnovu spiska sa brojem bodova svakog takmičara prikazuje ispis kao što je dat iznad, ali koji će raditi brže od trenutnog i moći odgovoriti na hiljade zahtjeva u sekundi.

Ulazni podaci

Na standardnom ulazu (tastatura) nalaze se ulazni podaci u sljedećem formatu.

Najprije se unosi pozitivan cijeli broj *n* koji predstavlja ukupan broj takmičara.

Narednih *n* redova ulaza su podaci o takmičaru i to: šifra takmičara, a zatim broj bodova koje je takmičar ostvario razdvojeni razmakom. Šifra je tekst sastavljen od velikih i malih slova i cifara, ne duži od 30 karaktera, pretpostavite da se šifre neće ponavljati. Broj bodova je cijeli broj na intervalu [0,1000]. Takmičari nisu poredani nikakvim posebnim redoslijedom.

Nakon podataka o takmičarima slijede zahtjevi koji pristižu putem web stranice, pri čemu je svaki zahtjev predstavljen kao šifra takmičara u zasebnom redu. Sa ulaza se učitava cijeli broj k koji predstavlja broj zahtjeva, a zatim slijede zahtjevi koji se mogu ponavljati, a moguće je i da se pojave nepostojeće šifre.

Izlazni podaci

Program treba na standardni izlaz (ekran) ispisati odgovore na zahtjeve onim redom kojim su zahtjevi dati na ulazu. Svaki odgovor treba biti u zasebnom redu. Format izlaza je dat u zadatku. U slučaju da je navedena nepostojeća šifra treba ispisati poruku "Nepoznata sifra." (bez navodnika, sa tačkom na kraju).

U slučaju da su dva ili više takmičara osvojili isti broj bodova, svi se nalaze na istom mjestu i to onom koje je povoljnije za takmičara.

Ograničenja

Vaš program ne smije raditi duže od 10s i ne smije koristiti više od 1 MiB memorije. U datoteci rezultati.txt nalaziće se rezultati za najviše 100.000 takmičara. U datoteci zahtjevi.txt nalaziće se najviše 100.000 zahtjeva.

50% testova sastojaće se od datoteka sa najviše 1.000 takmičara i najviše 1.000 zahtjeva. 20% testova će imati najviše 1.000 takmičara, ali može imati više od 1.000 zahtjeva. 20% testova će

imati najviše 1.000 zahtjeva, ali može imati više od 1.000 takmičara. Konačno, 10% testova će imati više od 1.000 takmičara i više od 1.000 zahtjeva.

Primjer ulaza i izlaza

STANDARDNI ULAZ:

```
4
Agunimon 521
DukeNukem 782
Charmander 0
Eevee 521
6
Agunimon
DukeNukem
Agunimon
Eevee
Pikachu
Charmander
```

STANDARDNI IZLAZ:

```
Osvojili ste 521 bodova. Nalazite se na 2 mjestu od 4 takmicara.
Osvojili ste 782 bodova. Nalazite se na 1 mjestu od 4 takmicara.
Osvojili ste 521 bodova. Nalazite se na 2 mjestu od 4 takmicara.
Osvojili ste 521 bodova. Nalazite se na 2 mjestu od 4 takmicara.
Nepoznata sifra.
Osvojili ste 0 bodova. Nalazite se na 4 mjestu od 4 takmicara.
```

Objašnjenje izlaza: Najprije se ispisuju podaci za takmičara Agunimon (521 bod, 2. mjesto), zatim DukeNukem, pa ponovo Agunimon, pa Eevee itd. onim redom kojim su navedeni. Šifra Pikachu je nepoznata.

Rimska aritmetika (rimski)

Potrebno je napraviti program koji omogućuje obavljanje različitih aritmetičkih operacija nad rimskim brojevima: sabiranje, oduzimanje, množenje i dijeljenje. Pošto klasičnim rimskim brojevima nije moguće predstaviti negativne, decimalne brojeve, nulu ili broj veći od 3999, program pretpostavlja da će rezultat svih aritmetičkih operacija biti pozitivan cijeli broj na intervalu [1,3999].

Ulazni podaci

Na standardnom ulazu (tastatura) se nalazi niz aritmetičkih operacija sa rimskim brojevima. U prvoj liniji ulaza nalazi se broj *n* aritmetičkih izraza. Zatim se u svakoj narednoj liniji nalazi jedan izraz koji je uvijek oblika:

[OPERAND][OPERATOR][OPERAND]

Između operanda i operatora nema razmaka. Operator je jedan od:

+ sabiranje
- oduzimanje
* množenje
/ dijeljenje
% modulo (ostatak pri cjelobrojnom dijeljenju)
^ stepen

Operandi su rimski brojevi. U pitanju su stringovi sastavljeni od sljedećih rimskih cifara (biće korištena isključivo velika slova):

I	V	X	L	C	D	M
1	5	10	50	100	500	1000

Brojevi se dobijaju kombinovanjem cifara koje su navedene od većih ka manjim npr.

2637 = MMDCXXXVII

Izuzetak su cifre 4 i 9 koje se dobijaju po sljedećoj tabeli:

IV	IX	XL	XC	CD	CM
4	9	40	90	400	900

Sve operacije date na ulazu će biti takve da je njihov rezultat pozitivan cijeli broj u opsegu [1,3999].

Izlazni podaci

Na standardnom izlazu (ekran) potrebno je ispisati rezultate za *n* aritmetičkih operacija, u vidu rimskih brojeva. Svaki rezultat se treba nalaziti na zasebnoj liniji izlaza.

Ograničenja

Vaš program ne smije raditi duže od 0.1s i ne smije koristiti više od 16 MiB memorije.

Primjer ulaza i izlaza

STANDARDNI ULAZ:

```
4
XXVII+CXXIII
C-I
L%III
XVI^II
```

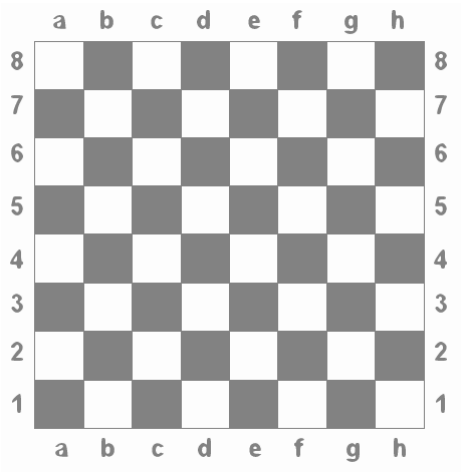
STANDARDNI IZLAZ:

```
CL
XCIX
II
CCLVI
```

Konj i riža (konj_i_riza)

Na šahovsku tablu (dimenzija 8x8) prosuta je riža. U svakom polju se našao određeni broj zrna riže. Šahovska figura konj treba da u nekom datom broju koraka *k* skupi najveći mogući broj zrna riže, ali se pri tome može kretati po šahovskoj tabli samo na način kako se inače ta šahovska figura kreće u igri šaha.

Vaš zadatak je da napravite program koji za neki početni raspored zrna riže i broj koraka k ispisuje put kojim konj treba preći. Zapamtite da su polja na šahovskoj tabli označena slovima a, b, c,..., h i brojevima 1,2,3,...,8 kao na slici ispod.



Konj kreće iz polja a1 (donji lijevi ugao table) u kojem nema zrna. Konj može više puta posjetiti isto polje, ali broj zrna za to polje se ubraja u sumu samo jednom.

Ulazni podaci

Na standardnom ulazu (tastatura) najprije se nalazi jedan cijeli broj *k* koji označava broj koraka koje smije napraviti konj. Zatim u narednih 8 redova slijedi po 8 cijelih brojeva razdvojenih razmakom koji predstavljaju broj zrna u svakom polju table i to redom kao na slici, dakle:

a8 b8 c8 ... h8

a7 b7 c7 ... h7

...

Prema postavci zadatka u polju a1 broj zrna će biti 0, nije potrebno to provjeravati.

Izlazni podaci

Program treba na standardni izlaz (ekran) ispisati broj zrna koja je konj skupio, a u sljedećem redu put kojim je konj prošao da bi ta zrna skupio. Put treba biti predstavljen stringom koji navodi polja kojima se konj kretao razdvojena razmakom, npr.

a1 b3 d4 e2

Kreće se iz polja a1 koje uvijek mora biti navedeno, a ne ubraja se u broj koraka *k*, npr. za put dat iznad *k*=3.

Ograničenja

Vaš program ne smije raditi duže od 0.1s i ne smije koristiti više od 16 MiB memorije. Broj koraka *k* neće biti veći od 100.

(primjer ulaza i izlaza na sljedećoj stranici)

Primjer ulaza i izlaza

STANDARDNI ULAZ:

```
3
0 0 99 0 0 0 0 0
0 0 0 0 0 0 0 99
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 1 0 5 0 0 0 0
0 2 0 0 0 1 0 0
0 0 3 3 0 0 0 0
0 0 0 0 0 10 0 0
```

STANDARDNI IZLAZ:

```
15
a1 b3 d2 f1
```

Objašnjenje izlaza: Uz ograničenje da konj može napraviti najviše tri skoka, može se skupiti 15 zrna riže i to ako se slijedi put a1 - b3 - d2 - f1 (2 + 3 + 10 = 15). Konj ne može nikako u tri skoka doći do polja sa 99 zrna riže.

Šest koraka (sest_koraka)

1929. godine mađarski pisac Frigyes Karinthy postavio je poznatu hipotezu "šest koraka" ili "šest stepeni udaljenosti" (six degrees of separation). Da bi pokazao kako je svijet ustvari mali, Karinthy je pretpostavio da se između bilo koje dvije osobe na svijetu može uspostaviti lanac prijatelja takav da on nije duži od 6 koraka. Primjer lanca od 6 koraka je recimo ako osoba A poznaje osobu B, B poznaje C, C poznaje D, D poznaje E, E poznaje F, a F poznaje G. Veza između dvije osobe funkcioniše u oba smjera, znači ako osoba A poznaje osobu B, pretpostavljamo i da osoba B poznaje osobu A odnosno da to ne treba posebno navoditi.

Potrebno je da napravite program koji će provjeriti ovu hipotezu na osnovu podataka sa neke društvene mreže.

Ulazni podaci

Na standardnom ulazu (tastatura) nalaze se ulazni podaci u sljedećem formatu.

Najprije je dat pozitivan cijeli broj *n* koji predstavlja broj parova prijatelja. Zatim slijedi niz od *n* parova osoba koje su prijatelji, oblika [IME] [PREZIME] [IME] [PREZIME]. Pretpostavite da su imena i prezimena stringovi sastavljeni isključivo od velikih i malih slova engleskog alfabeta, dakle bez razmaka, crtica itd. Svaki par se nalazi u zasebnoj liniji, a imena i prezimena su razdvojena znakom razmak.

Nakon *n* parova prijatelja slijede parovi osoba za koje treba provjeriti hipotezu. Unosi se pozitivan cijeli broj *k* koji predstavlja broj parova za provjeru. Zatim slijed niz od *k* parova osoba za koje treba izračunati dužinu lanca prijatelja, te ispisati da li je hipoteza ispunjena ili nije ispunjena.

Izlazni podaci

Na standardnom izlazu (ekran) potrebno je ispisati tekst "hipoteza ispunjena" (bez navodnika, znakova interpunkcije i sl.) ako je dužina lanca prijatelja između te dvije osobe manja ili jednaka 6, odnosno "hipoteza nije ispunjena" ako je lanac duži od 6 koraka ili ako uopšte ne postoji lanac prijatelja između te dvije osobe. Ove poruke treba ispisati onim redom kojim su parovi navedeni na ulazu.

Ograničenja

Vaš program ne smije raditi duže od 1s i ne smije koristiti više od 16 MiB memorije. Broj *n* parova prijatelja kao i broj *k* parova za provjeru neće biti veći od 1.000. U 20% testnih slučajeva brojevi *n* i *k* će biti manji od 20.

Primjer ulaza i izlaza

STANDARDNI ULAZ:

```
13
Ahmed Ahmedic Berina Berinic
Berina Berinic Selma Selmic
Selma Selmic Kristina Kristic
Predrag Predic Kristina Kristic
Osman Osmanovic Predrag Predic
Osman Osmanovic Nermina Nermic
Nermina Nermic Zaim Zaimovic
Zaim Zaimovic Mustafa Mustafic
Ahmed Ahmedic Sasa Sasic
Mustafa Mustafic Sasa Sasic
```

```
Leopold Ludwig Sasa Sasic
Hans Schultz Leopold Ludwig
Neznanko Neznankovic Imenko Prezimenkovic
4
Mustafa Mustafic Ahmed Ahmedic
Kristina Kristic Leopold Ludwig
Imenko Prezimenkovic Predrag Predic
Hans Schultz Predrag Predic
```

STANDARDNI IZLAZ (ekran):

```
hipoteza ispunjena
hipoteza ispunjena
hipoteza nije ispunjena
hipoteza nije ispunjena
```

Objašnjenje izlaza:

1. Između osoba "Mustafa Mustafic" i "Ahmed Ahmedic" postoje dva lanca, jedan od njih ima 7 koraka (Ahmed - Berina - Selma - ... - Mustafa) a drugi samo 2 koraka (Ahmed - Saša - Mustafa).

2. Između osoba "Kristina Kristic" i "Leopold Ludwig" postoji lanac od 6 koraka.

3. Između osoba "Imenko Prezimenkovic" i "Predrag Predic" uopšte ne postoji lanac jer su Imenko i Neznanko prijatelji isključivo jedan drugom i ne poznaju više nikoga.

4. Između osoba "Hans Schultz" i "Predrag Predic" postoje dva lanca, no oba imaju po 7 koraka tako da hipoteza nije ispunjena.

PRIJEDLOG RJEŠENJA (C++)

Takmičenje

```
#include <iostream>
#include <algorithm>
#include <fstream>
#include <vector>
#include <string>

using namespace std;

struct rezultat {
	string sifra;
	int bodovi, mjesto;
};


bool sort_po_bodovima(rezultat a, rezultat b){ return a.bodovi>b.bodovi; }
bool sort_abecedno(rezultat a, rezultat b){ return a.sifra<b.sifra; }

int main() {
	vector<rezultat> rezultati;
	vector<string> sifre;

	// Ucitavanje
	int br_rezultata;
	cin >> br_rezultata;
	for (int i(0); i<br_rezultata; i++) {
		struct rezultat r;
		cin >> r.sifra >> r.bodovi;
		rezultati.push_back(r);
		sifre.push_back(r.sifra);
	}

	// Sortiranje po broju bodova
	sort(rezultati.begin(), rezultati.end(), sort_po_bodovima);
	// Odredjivanje mjesta
	for (int i(0); i<br_rezultata; i++) {
		rezultati[i].mjesto = i+1;
		if (i>0 && rezultati[i].bodovi == rezultati[i-1].bodovi)
rezultati[i].mjesto = rezultati[i-1].mjesto;
	}
	// Sortiranje po sifri
	sort(rezultati.begin(), rezultati.end(), sort_abecedno);
	sort(sifre.begin(), sifre.end());

	// Procesiranje zahtjeva
	int br_zahtjeva;
	cin >> br_zahtjeva;
	for (int i(0); i<br_zahtjeva; i++) {
		string zahtjev;
		cin >> zahtjev;
		if (zahtjev == "") continue;
		std::vector<string>::iterator low,up;
		low = upper_bound(sifre.begin(), sifre.end(), zahtjev);
		if (low == sifre.begin() || *(low-1) != zahtjev) cout << "Nepoznata
sifra.\n";
		else {
			int idx = low - sifre.begin() - 1;
			//cout << "idx "<<idx<<"\n";
			cout << "Osvojili ste "<<rezultati[idx].bodovi<<" bodova.
Nalazite se na "<<rezultati[idx].mjesto<<" mjestu od "<<br_rezultata<<"
```

```
takmicara.\n";
                }
        }

        return 0;
}
```

Rimski

```
#include <iostream>
#include <cmath>

using namespace std;

string rimcifre[13] = { "M", "CM", "D", "CD", "C", "XC", "L", "XL", "X", "IX",
"V", "IV", "I" };
int arapcifre[13] = { 1000, 900, 500, 400, 100, 90, 50, 40, 10, 9, 5, 4, 1};

int rimski2arapski(string rimski)
{
        int k(0);
        int arapski(0);
        for (int i(0); i<13; i++) {
                while (rimski.substr(k, rimcifre[i].length()) == rimcifre[i]) {
                        arapski += arapcifre[i];
                        k += rimcifre[i].length();
                }
        }
        return arapski;
}

string arapski2rimski(int arapski) {
        string rimski("");
        for (int i(0); i<13; i++) {
                while (arapski>=arapcifre[i]) {
                        arapski -= arapcifre[i];
                        rimski += rimcifre[i];
                }
        }
        return rimski;
}

int main() {
      int n;
      cin >> n;
      for (int i(0); i<n; i++) {
            string operacija;
            cin >> operacija;
            int j(0);
            while (operacija[j] >= 'C' && operacija[j] <= 'X') j++;
            int operand1 = rimski2arapski(operacija.substr(0,j));
            int operand2 = rimski2arapski(operacija.substr(j+1));
            int rez;
            switch (operacija[j]) {
                  case '+':
                        rez=operand1+operand2;
                        break;
                  case '-':
                        rez=operand1-operand2;
                        break;
                  case '*':
                        rez=operand1*operand2;
                        break;
                  case '/':
                        rez=operand1/operand2;
                        break;
                  case '%':
                        rez=operand1%operand2;
```

```
                        break;
                  case '^':
                        rez=pow(operand1, operand2);
                        break;
            }
            cout << arapski2rimski(rez) << endl;
      }
      return 0;
}
```

Konj i riža

```
#include <iostream>
#include <vector>
#include <string>

using namespace std;

struct stanje {
	vector<int> ploca, put;
	int riza;
};

vector<int> moguci_skokovi(int pozicija) {
	vector<int> rezultat;
	int x=pozicija/8;
	int y=pozicija%8;
	if (x>1) {
		if (y>0) rezultat.push_back((x-2)*8+y-1);
		if (y<7) rezultat.push_back((x-2)*8+y+1);
	}
	if (x>0) {
		if (y>1) rezultat.push_back((x-1)*8+y-2);
		if (y<6) rezultat.push_back((x-1)*8+y+2);
	}
	if (x<7) {
		if (y>1) rezultat.push_back((x+1)*8+y-2);
		if (y<6) rezultat.push_back((x+1)*8+y+2);
	}
	if (x<6) {
		if (y>0) rezultat.push_back((x+2)*8+y-1);
		if (y<7) rezultat.push_back((x+2)*8+y+1);
	}
	return rezultat;
}

void ispisi_pos(int pos) {
	int x = pos/8;
	int y = pos%8;
	cout << char('a'+y) << (x+1);
}

void ispisi_put(vector<int> put) {
	for (int i(0); i<put.size(); i++) {
		ispisi_pos(put[i]);
		cout << " ";
	}
	cout << endl;
}

stanje dfs(stanje s, int max_koraka) {
	int pozicija = s.put[s.put.size()-1];
	vector<int> m = moguci_skokovi(pozicija);
	stanje najbolje = s;
	for (int i(0); i<m.size(); i++) {
		stanje novo = s;
		novo.put.push_back(m[i]);
		novo.riza += novo.ploca[m[i]];
		novo.ploca[m[i]] = 0;
		if (novo.put.size() <= max_koraka)
			novo = dfs(novo, max_koraka);
```

```
            if (novo.riza >= najbolje.riza) najbolje = novo;
      }
      return najbolje;
}

int main() {
      stanje pocetno;
      int max_koraka;
      cin >> max_koraka;

      // Okrecemo plocu tako da je a1 = 0
      pocetno.ploca.resize(64);
      for (int i(0); i<8; i++)
            for (int j(0); j<8; j++) {
                  int pos = (7-i)*8 + j;
                  cin >> pocetno.ploca[pos];
            }

      pocetno.put.push_back(0);
      pocetno.riza=0;

      stanje najbolje = dfs(pocetno, max_koraka);
      cout << najbolje.riza << endl;
      ispisi_put(najbolje.put);
      return 0;
}
```

Šest koraka

```
#include <iostream>
#include <vector>
#include <string>
#include <iomanip>
#include <queue>

using namespace std;

vector<string> osobe;
vector<vector<bool> > matricaSusjedstva;
vector<bool> posjeceno;

// Funkcija koja traĹľi osobu u grafu, a ako je ne naÄ‘e dodaje je u graf
int traziOsobu(string ime) {
	for (int i(0); i<osobe.size(); i++)
		if (osobe[i] == ime) return i;

	// Dodajemo osobu u graf
	osobe.push_back(ime);
	// Trebamo se pobrinuti da je matrica susjedstva popunjena sa false
	matricaSusjedstva.resize(osobe.size(), vector<bool>(osobe.size(), false));
	for (int i(0); i<osobe.size()-1; i++)
		matricaSusjedstva[i].resize(osobe.size(), false);
	return osobe.size()-1;
}

int bfs(int p, int q, int maxdubina) {
	queue<int> putevi;
	putevi.push(p);

	posjeceno[p] = true;
	int dubina=1;
	int dodano_ova_dubina=1, dodano_sljedeca_dubina=0;

	while (!putevi.empty()) {
		int cvor=putevi.front();
		putevi.pop();
		for (int i(0); i<matricaSusjedstva.size(); i++) {
			if (i==cvor) continue;
			if (matricaSusjedstva[cvor][i] && !posjeceno[i]) {
				if (i==q) return dubina;
				putevi.push(i);
				dodano_sljedeca_dubina++;
				posjeceno[i] = true;
			}
		}
		dodano_ova_dubina--;
		if (dodano_ova_dubina == 0) {
			dodano_ova_dubina=dodano_sljedeca_dubina;
			dodano_sljedeca_dubina=0;
			dubina++;
			if (dubina>maxdubina) return -1;
		}
	}

	return -1;
}
```

```
int main() {
	int n, k;
	cin >> n;

	// Kreiraj graf prijatelja
	for (int i(0); i<n; i++) {
		string ime1, prezime1, ime2, prezime2;
		cin >> ime1 >> prezime1 >> ime2 >> prezime2;

		int idx1 = traziOsobu(ime1 + " " + prezime1);
		int idx2 = traziOsobu(ime2 + " " + prezime2);
		// Ako je A prijatelj B onda je B prijatelj A
		matricaSusjedstva[idx1][idx2] = true;
		matricaSusjedstva[idx2][idx1] = true;
	}

	// Provjera hipoteze
	cin >> k;
	for (int i(0); i<k; i++) {
		string ime1, prezime1, ime2, prezime2;
		cin >> ime1 >> prezime1 >> ime2 >> prezime2;

		int idx1 = traziOsobu(ime1 + " " + prezime1);
		int idx2 = traziOsobu(ime2 + " " + prezime2);

		posjeceno.resize(osobe.size());
		for (int j(0); j<osobe.size(); j++) posjeceno[j]=false;

		int brojKoraka = bfs(idx1, idx2, 6);
		//cout << brojKoraka << " ";

		if (brojKoraka <= 6 && brojKoraka != -1)
			cout << "hipoteza ispunjena\n";
		else
			cout << "hipoteza nije ispunjena\n";
	}

	return 0;
}
```