Korešpondenčný seminár z programovania
XIX. ročník, 2001/2002
Katedra vyučovania informatiky FMFI UK,
Mlynská Dolina, 842 48 Bratislava

KSP finančne podporuje nadácia Open Society Fund Bratislava
IUVENTA – zariadenie pre voľný čas detí, mládeže a dospelých
MICROSTEP spol. s.r.o.

Príklady 2. kola zimnej časti

Milé deti,
prikosnilo sa! A ešte len uvidíte, ako bude, keď sa tetka Perinbabka potkne a na svoju perinu spadne...

Preto sa uvelebte do postele, prikryte sa perinou a vezmite si tieto zadania KSP do ruky. Keď vás začne omíňať ľavý bok, ľahnite si na pravý.

Svoje riešenia nám posielajte do **17. decembra**, a nezabudnite priložiť známky v hodnote 15 Sk,-.

Kos si, kosa, kos, umrzne ti nos!

KSPáci

1. Ochrana zdravia pri práci

Kontrolná Skupina Pracovníkov na svojom nedávnom zasadnutí rozhodla, že vo všetkých prevádzkach s viac ako 10 zamestnancami treba zvýšiť bezpečnostné opatrenia. Dôvodom takéhoto rozhodnutia bol vysoký počet úrazov v pracovnej dobe. Preto musí kontrolná skupina vykonať viacero testov, na základe ktorých by mala určiť optimálne bezpečnostné opatrenia.

Prvý problém sa zjavil hneď na začiatku: ako vybrať vhodných kandidátov na jednotlivé testy? Testy sú pomerne drahé a preto treba zo všetkých zamestnancov vybrať nejakých k, ale nie akýchkoľvek – vybraný zamestnanec musí patriť svojim vekom čo najviac do „stredu“, t.j. jeho vek sa musí (v absolútnej hodnote) čo najmenej odlišovať od stredného veku[1] všetkých zamestnancov.

Úloha: Skúste pomôcť našej Kontrolnej Skupine Pracovníkov a napíšte im program, ktorý z danej množiny N pracovníkov vyberie K pracovníkov, ktorých vek sa čo najmenej líši od strednej hodnoty veku všetkých pracovníkov. Stredná hodnota veku je $\lfloor \frac{n}{2} \rfloor$-tý vek po utriedení od najmenšieho po najväčší.

Snažte sa, aby váš program pracoval čo najrýchlejšie, pracovníkov je veľmi veľa a peňazí bude možno tiež dosť...

Príklad:

Vstup:
$N = 12$, $K = 4$
9, 13, 11, 10, 11, 15, 15, 14, 12, 9, 5, 7

Výstup:
Pracovníci: 3, 4, 5, 9
(ich veky v poradí sú 11, 10, 11, 12)

2. O ťavích dostihoch

Neviem, či to viete, ale čoskoro sa budú konať veľké ťavie dostihy. Ak ste náhodou na takých ťavích dostihoch ešte neboli a neviete, ako vyzerajú, tak vám ich v krátkosti popíšem. Dostihov sa zúčastňuje N tiav. Je daná dráha, ktorú majú zabehnúť. Pred začiatkom dostihov

[1] občas sa to zvykne nazývať aj medián

stoja ťavy na štarte kupodivu nie vedľa seba na jednej línii, ale jedna za druhou. Podľa toho, v akom poradí stoja, sú aj očíslované. Tá, čo je najbližšie k cieľu, má číslo 1, tá za ňou 2, atď. Po odštartovaní ťavy utekajú čo najrýchlejšie, ako vedia, aby sa dostali čo najskôr do cieľa. Výsledky dostihov sa určia podľa toho, v akom poradí dobehli ťavy do cieľa. Tá, čo dobehla najskôr je prvá, tá, čo dobehla po nej je druhá, atď. Občas sa počas dostihov stane, že nejaká ťava predbehne inú. Keďže ťavy sú veľmi vytrvalé, nikdy sa nestane, že ťava predbehne ťavu, ktorá ju predtým (samozrejme počas toho istého závodu) predbehla.

Ako to už býva na ťavích dostihoch zvykom, môžete uzatvárať rôzne stávky. Tento rok chce stávková spoločnosť zaviesť nový druh stávok. Budete si môcť napríklad staviť, že počas dostihov nastane práve K predbehnutí. Aby vedeli stanoviť kurzy takýchto stávok potrebujú vedieť, koľkými spôsobmi môže daný počet predbehnutí nastať.

Úloha: Sú dané kladné celé čísla N a K. Určte, koľkými rôznymi spôsobmi môžu prebiehať dostihy tak, že nastane práve K predbehnutí. Dva spôsoby považujeme za rovnaké, ak vedú k rovnakému výslednému usporiadaniu tiav.

Príklad:

Vstup:	**Výstup:**
$N = 3$	2
$K = 2$	*(sú to: 2 3 1, 3 1 2)*

3. O Martowovi II

Po siedmych ročníkoch a jednom kole prišiel mimozemšťan Martow opäť len tak mimochodom na Fakultu Matematiky, Fyziky a Informatiky a doniesol študentom d-rozmerný plán mestskej hromadnej dopravy Manhattanu.

Plán bol veľmi jednoduchý, boli na ňom nakreslené len malé modré bodky. Ako im Martow vysvetlil tie bodky sú autobusové zástavky. Medzi každými dvoma zástavkami chodí jeden autobus. Lenže tie autobusy nechodia len tak, ale chodia len v smere jednej z d svetových strán.[2] Ulice v Manhattane totiž vedú len týmito smermi. Samozrejme autobus môže kedykoľvek odbočiť a pobrať sa inou svetovou stranou. Martow by chcel vedieť, ktorý autobus má najdlhšiu okružnú trasu. Pomôžte mu!

Úloha: Je dané d – počet rozmerov mapy, N – počet zástavok. Ďalej má každá zástavka daných d svojich súradníc $(x_1, x_2, \ldots, x_d)$. Čas, ktorý trvá prejsť autobusu zo zástavy so súradnicami $(x_1, x_2, \ldots, x_d)$ na zástavku so súradnicami $(y_1, y_2, \ldots, y_d)$ je $|x_1 - y_1| + |x_2 - y_2| + \ldots + |x_d - y_d|$. Zistite dvojicu zástavok, prejsť medzi ktorými to trvá autobusu najdlhšie.

Poznámka: Počet zástavok môže byť veľmi veľký v porovnaní s počtom rozmerov.

Príklad:

Vstup:	**Výstup:**

$d = 4$, $N = 5$
`1 2 3 4`
`6 7 4 5`
`9 0 -1 3`
`3 5 8 4`
`4 11 0 0`
	Najdlhšie to trvá medzi (9,0,-1,3) a (3,5,8,4).

4. O podnikovej sieti

V istom nemenovanom veľkom podniku majú veľa počítačov. A majú s nimi veľké problémy. Veď posúďte sami: ledva stihli naučiť sekretárky behať s batohmi diskiet od počítača k

[2] Napr. pre tri rozmery chodia autobusy len hore a dole, vľavo a vpravo a dopredu a dozadu.

počítaču, keď tu zrazu vedenie podniku rozhodlo, že dajú spraviť počítačovú sieť. Dali kúpiť do každého počítača niekoľko sieťových kariet, kúpili kopu káblov a... zostali sa na ne pozerať, nevediac, čo s nimi. Potom niekoho napadla spásonosná myšlienka – veď elektrikár Braňo by sa predsa mal vyznať do počítačov! Tak ho zavoláme, nech nám spraví sieť. Mal by pospájať počítače tak, aby sa po sieti dalo komunikovať medzi ľubovoľnými dvoma počítačmi.

Keď Braňo prišiel, hneď videl, že to s tou sieťou nebude také jednoduché. Na všetkých počítačoch v podniku totiž beží operačný systém WOK-ná[3] XP (eXtrémne Padavý – padne, akonáhle sa mu niečo znepáči). A WOK-ná XP okrem iného vyžadujú nasledovné podmienky:

- Z každej sieťovej karty v počítači musí vychádzať práve 1 kábel
- Žiaden počítač nesmie byť spojený sám so sebou
- Každé dva počítače môžu byť priamo prepojené najviac 1 káblom.

V opačnom prípade WOK-ná XP okamžite padnú. A navyše vedenie podniku samozrejme nechce počuť ani slovo o tom, že by sa z nejakého počítača mala nejaká sieťová karta vybrať. Veď za ne dali toľké peniaze!

A tak Braňo dlho smutne sedel nad hromadou káblov a premýšľal. Teraz (o tri dni neskôr) je vyhladnutý, vysmädnutý a presvedčený, že sa to nedá. A rád by to vedeniu dokázal, lebo ináč ho vyhodia.

Úloha: Na vstupe je počet počítačov N a pre každý z nich, koľko má sieťových kariet (inými slovami s koľkými inými počítačmi ho Braňo má spojiť). Napíšte program, ktorý zistí, či sa dajú počítače poprepájať káblami tak, aby operačný systém nepadol a dalo sa po sieti komunikovať medzi každými dvoma počítačmi. Káblov má Braňo dosť.

Príklad:

Vstup:
$N = 5$
3 3 2 1 1

Výstup:
`Áno` *(Spojí napr. dvojice 1–2, 1–3, 1–4, 2–3, 2–5)*

Vstup:
$N = 5$
6 2 2 2 2

Výstup:
`Nie` *(Počítač 1 by mal byť spojený so 6 inými, to sa ale zjavne nedá)*

Vstup:
$N = 4$
1 1 1 1

Výstup:
`Nie` *(Nech ich spojíme akokoľvek, súvislú sieť nespravíme)*

5. O čiernych krabičkách II

Po nákupe veľkokapacitných pamätí typu Čierna skrinka v.1.0 panovala niekoľko dní v softvérovom družstve SoDr spokojnosť. Čoskoro sa však na trhu objavila vylepšená verzia Čiernej skrinky – Čierna skrinka v.2.0.

Čierna skrinka v.2.0 používa tieto tri príkazy:

- `procedure Insert(dest,val);` – uloží do čiernej skrinky `dest` hodnotu `val`.
- `function ExtractMin(dest);` – vyberie z čiernej skrinky `dest` najmenšiu hodnotu, ktorá sa v nej nachádza. Túto hodnotu funkcia zmaže z pamäte a vráti ako návratovú hodnotu.
- `function Empty(dest);` – vráti `TRUE`, ak je čierna skrinka `dest` prázdna, inak vráti `FALSE`.

[3] Nemýliť si s wokom, wok je taká čínska panvica. WOK-ná je záhorácka lokalizácia operačného systému Wardrobe O.K.

Príkaz `Empty` sa uskutočňuje v jednotkovom čase, všetky ostatné príkazy sa uskutočňujú v čase $O(\lg n)$, kde n je počet hodnôt uložených v pamäti.

Príklad použitia pamäte:

Príkaz:	**Návratová hodnota**	**Obsah pamäte po vykonaní príkazu:**
Insert(a,1)		1
Insert(a,5)		1,5
Insert(a,3)		1,5,3
Empty(a)	FALSE	1,5,3
ExtractMin(a)	1	5,3
ExtractMin(a)	3	5
ExtractMin(a)	5	
Empty(a)	TRUE	

V softvérovom družstve SoDr dostali výhodnú ponuku na upgrade pamäte Čierna skrinka z verzie v1.0 na verziu v2.0. Ponuka bola taká výhodná, že ju nemohli odmietnuť a všetky pamäte upgradovali. Potrebovali by však pamäť, z ktorej by sa dala vyberať najmenšia aj najväčšia hodnota, ktorá sa v nej nachádza. Budú preto potrebovať vašu pomoc.

Úloha: Napíšte program, ktorý bude emulovať pamäť, z ktorej možno vyberať najmenšiu aj najväčšiu hodnotu. Program má načítavať a spracovávať tieto príkazy:

- `procedure Insert(val);` – uloží do pamäte hodnotu `val`
- `function ExtractMin;` – vyberie z pamäte najmenšiu hodnotu, ktorá sa v nej nachádza. Túto hodnotu funkcia zmaže z pamäte a vráti ju ako návratovú hodnotu.
- `function ExtractMax;` – vyberie z pamäte najväčšiu hodnotu, ktorá sa v nej nachádza. Túto hodnotu funkcia zmaže z pamäte a vráti ju ako návratovú hodnotu.
- `function Empty;` – vráti `TRUE`, ak je pamäť prázdna, inak vráti `FALSE`

Váš program môže používať konštantný počet premenných typu Čierna skrinka v2.0. Okrem nich môže používať len konštantný počet iných premenných.

Snažte sa, aby váš program pracoval efektívne, teda aby príkazy vykonával čo najrýchlejšie a aby celkový počet dát uložených v premenných typu Čierna skrinka bol čo najmenší.

Príklad:

Vstup:	**Výstup:**
Put(1)	
Put(3)	
Put(5)	
Empty	FALSE
ExtractMax	5
ExtractMin	1
ExtractMin	3
Empty	TRUE