Na řešení úloh máte 4,5 hodiny čistého času.

Řešení každého příkladu musí obsahovat:

- **Popis řešení,** to znamená slovní popis použitého algoritmu, argumenty zdůvodňující jeho správnost (případně důkaz správnosti algoritmu), diskusi o efektivitě vašeho řešení (časová a paměťová složitost). Slovní popis řešení musí být jasný a srozumitelný i bez nahlédnutí do samotného zápisu algoritmu (do programu).
- **Program.** V úlohách **P-III-1** a **P-III-2** je třeba uvést dostatečně podrobný zápis algoritmu, nejlépe ve tvaru zdrojového textu nejdůležitějších částí programu v jazyce Pascal nebo C. Ze zápisu můžete vynechat jednoduché operace jako vstupy, výstupy, implementaci jednoduchých matematických vztahů apod. V úloze **P-III-3** uveďte příslušné programy pro registrové počítače.

Hodnotí se nejen správnost programu, ale také kvalita popisu řešení a efektivita zvoleného algoritmu.

P-III-1 Agenti

Jistá nejmenovaná tajná společnost má N agentů. Z důvodu utajení může každý agent vydávat rozkazy jen několika dalším agentům. Agent, který dostane rozkaz, pošle tento rozkaz všem agentům, jimž může vydávat rozkazy. Šéfem společnosti je takový agent, který když vydá rozkaz, tak ho časem dostanou všichni agenti. (Společnost může mít i více šéfů, případně nemusí mít žádného šéfa.)

Soutěžní úloha: Na vstupu je dán počet agentů N. Agenti jsou označeni čísly od 7 (přesněji 007) do $N + 6$. Pro každého agenta je také dán seznam agentů, kterým může vydat rozkaz. Navrhněte efektivní algoritmus, který určí šéfa tajné společnosti (pokud jich existuje více, stačí nalézt jednoho libovolného z nich) nebo zjistí, že tajná společnost žádného šéfa nemá.

Příklad 1:

Vstup:	*Výstup:*
$N = 3$	Šéfem je agent 7.
Agent 7 rozkazuje agentovi 8.	
Agent 8 rozkazuje agentovi 9.	
Agent 9 rozkazuje agentovi 7.	

Příklad 2:

Vstup:	*Výstup:*
$N = 4$	Žádný agent není šéfem.
Agent 7 nerozkazuje nikomu.	
Agent 8 nerozkazuje nikomu.	
Agent 9 rozkazuje agentům 7 a 8.	
Agent 10 rozkazuje agentům 7 a 8.	

P-III-2 Teploty

Meteorologická stanice měří každou minutu teplotu vzduchu. Meteorologové by potřebovali program, který by jim v každém okamžiku sděloval, jaká nejnižší teplota byla naměřena během posledních K minut. Vaším úkolem bude napsat tento program.

Na vstupu je dáno číslo K, následuje posloupnost naměřených teplot ukončená hodnotou -1000. Váš program musí po přečtení každé teploty ihned vypsat nejnižší z posledních K načtených teplot (resp. nejnižší ze všech načtených teplot, jestliže jich dosud bylo méně než K).

Příklad:

Vstup:	*Výstup:*
$K = 3$	
teploty:	
9.0	9.0
4.7	4.7
5.3	4.7
2.1	2.1
9.0	2.1
9.8	2.1
17.0	9.0
9.5	9.5
-1000	

P-III-3 Registrový počítač

V tomto kole se budeme zabývat tzv. *jednosměrnými registrovými počítači* – stejnými jako v domácím kole. Jejich formální definici najdete ve studijním textu, který následuje za zadáním soutěžní úlohy a je zcela shodný se studijním textem z domácího kola.

Soutěžní úloha:

a) Nechť R je řetězec tvořený písmeny `a`, `b`, `c`, `A`, `B`, `C`. Označme $m(R)$ řetězec tvořený malými písmeny obsaženými v R (ve stejném pořadí, v jakém se vyskytují v R). Analogicky označme $v(R)$ řetězec tvořený velkými písmeny v R. Řetězec $upcase(R)$ dostaneme z R tak, že nahradíme všechna malá písmena odpovídajícími velkými písmeny. Např. jestliže $R =$ `aaAcB`, potom $m(R) =$ `aac`, $v(R) =$ `AB` a $upcase(R) =$ `AAACB`.
Napište program pro jednosměrný registrový počítač, který bude řešit následující úlohu: Na vstupu dostane řetězec R tvořený písmeny `a`, `b`, `c`, `A`, `B`, `C`. Počítač ho má označit za správný právě tehdy, když $upcase(m(R)) = v(R)$. (Vyjádřeno slovně: když velká písmena obsažená v R tvoří „stejné“ slovo jako malá.)
Například vstupní řetězce `aA`, `Aa` a `abAcBaCABb` jsou tedy správné, zatímco řetězce `aa`, `BcbC` a `acACa` jsou špatné.
Váš program může použít libovolný konečný počet registrů, hodnotí se jen jeho správnost. Pokud si myslíte, že takový program neexistuje, dokažte to.

b) Dokažte, že pro libovolnou úlohu platí: Jestliže umíme sestrojit program pro registrový počítač, který řeší danou úlohu pomocí tří registrů, potom dokážeme sestrojit také program, který tuto úlohu řeší pomocí dvou registrů.
Jinými slovy: Ukažte postup, kterým lze libovolný existující program používající tři registry přepsat na ekvivalentní program, jenž potřebuje pouze dva registry. Nezapomeňte zdůvodnit správnost svého postupu.

Studijní text:

Registr je něco podobného jako proměnná. V registru může být uloženo *libovolně velké* nezáporné celé číslo. Na rozdíl od proměnných, které mezi sebou můžeme sčítat, odčítat a násobit, s registrem lze provádět jen tři jednoduché operace: zvětšit jeho obsah o 1, zmenšit jeho obsah o 1 (pokud se pokusíme zmenšit obsah registru obsahujícího hodnotu 0, zůstane v něm 0) a otestovat registr, zda je v něm 0. Na začátku výpočtu jsou ve všech registrech nuly.

Registrový počítač může používat neomezený počet registrů označených R_0, R_1, R_2, atd. Vedle registrů má k dispozici ještě *konečně velkou* pomocnou paměť.

Program pro registrový počítač budeme zapisovat v jazyce velmi podobném programovacímu jazyku Pascal. Programovací jazyk registrového počítače bude oproti Pascalu rozšířen například o příkazy pro práci s registry, naopak některé příkazy z Pascalu v něm budou zakázány.

Registrový počítač bude řešit úlohy následujícího typu: počítač dostane na vstupu zadáno slovo (řetězec písmen) a po nějakém čase odpoví, zda je toto slovo *správné* nebo *špatné*. Aby nám mohl odpovědět, zavedeme do programovacího jazyka speciální příkazy *Accept* a *Reject*. Jakmile se během výpočtu vykoná příkaz *Accept*, vstupní slovo je správné a výpočet končí. Jestliže se provede příkaz *Reject*, slovo je špatné a výpočet končí. Pokud se výpočet zacyklí nebo pokud skončí, aniž by se provedl příkaz *Accept* nebo *Reject*, zadané vstupní slovo je rovněž špatné.

Příkaz „přičti 1 k obsahu registru R“ budeme značit *Inc(R)*, „odečti 1 od obsahu registru R“ budeme značit *Dec(R)*. Výraz *Zero(R)* je pravdivý, jestliže je v registru R nula, v opačném případě je nepravdivý.

V každém programu můžeme použít jen konečně mnoho registrů. Kromě nich můžeme použít už jen konstantní počet pomocných proměnných typu *byte** (nemůžeme tedy používat pole!) a dále můžeme zvláštním způsobem využívat jednu speciální proměnnou *vstup* typu *char*. Obsah proměnné *vstup* lze měnit pouze provedením příkazu *Read(vstup)*. Jestliže počítač ještě nedočetl vstupní slovo, příkaz *Read(vstup)* z něj přečte jedno další písmeno a uloží ho do proměnné *vstup*. Pokud počítač již vstupní slovo dočetl, příkaz *Read(vstup)* uloží do proměnné *vstup* speciální znak `$`.

Jelikož registrový počítač má kromě registrů jen konečně mnoho paměti, nemůže si dovolit používat rekurzi (neměl by si kde pamatovat návratové adresy). My pro jistotu úplně zakážeme definovat a používat v programu procedury a funkce. Zakázáno je i volání všech standardních procedur a funkcí jazyka Pascal. V aritmetických výrazech lze používat pouze proměnné (tedy ne registry!), celočíselné konstanty, celočíselné operátory `+`, `-`, `*`, `div`, `mod` a závorky. V podmínkách se mohou používat výrazy $Zero(R_i)$, běžné relační operátory (`<`, `<=`, ...), logické spojky a závorky.

Z klíčových slov jazyka Pascal jsou tedy v programovacím jazyku registrového počítače povolena pouze následující: `var`, `begin`, `end`, `if`, `then`, `else`, `case`, `of`, `while`, `do`, `repeat`, `until`, `for`, `to`, `downto`, `div`, `mod`, `and`, `or`, `not`, `xor`.

Příklad 1:

Napište program pro jednosměrný registrový počítač, který bude řešit následující úlohu: Na vstupu je zadán řetězec písmen `a`, `b`, `c`. Nechť A označuje počet těch písmen `a` ve vstupním řetězci, za kterými už není žádné `c`. Podobně nechť B je počet `b`, za nimiž není žádné `c`. Počítač má vstupní řetězec označit za správný právě tehdy, když $A = B$.

Řešení: V registru R_1 si budeme pamatovat aktuální hodnotu A, v registru R_2 hodnotu B. Pokaždé, když přečteme ze vstupu písmeno `c`, oba registry vynulujeme. Na konci výpočtu jednoduše porovnáme hodnoty uložené v registrech. Rozmyslete si, že by stačilo použít jen jeden registr (v němž bychom měli hodnotu $|A - B|$).

```
var vstup:char;
begin
```

* taková proměnná obsahuje jedno celé číslo z rozmezí od 0 do 255.

```
   Read(vstup);
   while (vstup<>'$') do begin
      if (vstup='a') then Inc(R1);
      if (vstup='b') then Inc(R2);
      if (vstup='c') then begin
         while not Zero(R1) do Dec(R1);
         while not Zero(R2) do Dec(R2);
      end;
      Read(vstup);
   end;
   while not Zero(R1) do begin
      Dec(R1);
      if (Zero(R2)) then Reject;
      Dec(R2);
      end;
   if Zero(R2) then Accept;
end.
```

Příklad 2:

Napište program pro jednosměrný registrový počítač, který bude řešit následující úlohu. Na vstupu bude zadán řetězec písmen a. Počítač ho označí za správný právě tehdy, když je jeho délka mocninou tří.

Řešení: Přečteme vstupní slovo, přičemž si do R_1 uložíme jeho délku. Potřebujeme zjistit, zda je to mocnina tří. Uloženou hodnotu proto budeme dělit třemi, dokud to půjde. Jestliže nakonec dostaneme podíl 0 a zbytek 1, původní číslo bylo mocninou tří, jinak nebylo.

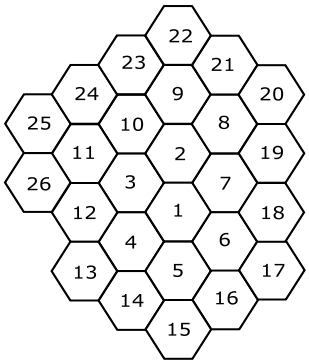
```
var vstup: char;
    zbytek: byte;
begin
   read(vstup);
   while (vstup<>'$') do begin Inc(R1); read(vstup); end;
   if Zero(R1) then Reject;
   while true do begin
      zbytek:=0;
      while not Zero(R1) do begin
         Dec(R1);
         zbytek:=(zbytek+1) mod 3;
         if (zbytek=0) then Inc(R2);
      end;
      if (Zero(R2) and (zbytek=1)) then Accept;
      if (zbytek<>0) then Reject;
      while not Zero(R2) do begin Dec(R2); Inc(R1); end;
   end;
end.
```

P-III-4 Psíci

Program: `psici.pas` / `psici.c` / `psici.cpp`
Vstup: `psici.in`
Výstup: `psici.out`

„Tak, a teď budete skákat vždycky, když zapískám. A každý jinak!“, rozkázal Konrád svým dvěma psíkům. Chudáci malí, musí teď oba poskakovat po louce tak dlouho, dokud se oba současně neschovají.

Svět je nekonečná šestiúhelníková síť. Políčka tvořící svět jsou očíslována přirozenými čísly počínaje od 1 po spirále. Louku tvoří prvních N políček světa. Na obrázku je příklad louky pro $N = 26$:



Na louce stojí naši dva psíci na políčkách S_1, S_2. Skrýš pro prvního psíka je na políčku T_1, pro druhého na políčku T_2. Na M políčkách louky rostou bodláky a psíci tam za žádnou cenu neskočí.

Na jedno písknutí přeskočí každý psík na libovolné sousední políčko, pokud na něm nerostou bodláky. Oba psíci nemohou nikdy skočit současně stejným směrem a nemohou také oba dopadnout najednou na stejné políčko. Při každém písknutí musí každý z nich přeskočit na jiné políčko (i kdyby už stál ve své skrýši).

Vaším úkolem je zjistit, na kolik nejméně písknutí se mohou oba psíci dostat současně do svých skrýší.

Vstup: Ve vstupním souboru `psici.in` následuje po sobě popis několika (maximálně pěti) problémů. Každý problém má na svém prvním řádku dvě čísla N ($2 \leq N \leq 500$), M ($0 \leq M \leq N - 2$), na druhém řádku čísla políček S_1, T_1, S_2, T_2 (v uvedeném pořadí, $1 \leq S_1, T_1, S_2, T_2 \leq N$). Následuje dalších M řádků s čísly políček, kde rostou bodláky. Vstupní soubor je ukončen řádkem obsahujícím dvě nuly ($M = N = 0$).

Můžete předpokládat, že vždy $S_1 \neq S_2$, $T_1 \neq T_2$ a že na políčkách S_1, S_2 nejsou bodláky. Na políčkách T_1 nebo T_2 bodláky být mohou – v takovém případě však úloha jistě nemá řešení.

Výstup: Do výstupního souboru `psici.out` zapište pro každý problém jeden řádek s nejmenším počtem písknutí, po němž mohou oba psíci současně stát ve svých skrýších. Pokud není možné tohoto výsledného stavu dosáhnout, vypište do výstupního souboru místo počtu písknutí řádek se slovem „nelze“.

Příklad:

Vstupní soubor `psici.in`

```
11 0
3 10 6 7
11 1
3 10 6 7
1
10 2
3 10 7 8
2
9
0 0
```

Výstupní soubor `psici.out`

```
2
3
nelze
```

P-III-5 AttoSoft

Program: `attosoft.pas` / `attosoft.c` / `attosoft.cpp`
Vstup: `attosoft.in`
Výstup: `attosoft.out`

Programátorské firmě AttoSoft se podařilo získat dalšího klienta, který potřebuje naprogramovat N programů. Vaškovi a jeho programátorům se však do práce moc nechtělo, a tak když přišel termín odevzdání, programy ještě stále nebyly hotové. Vašek se lekl a začal studovat smlouvu, kterou se zákazníkem podepsal.

Ve smlouvě byl pro každý program uveden vzorec, podle kterého se počítá pokuta za opožděné odevzdání programu v závislosti na délce zdržení. Naštěstí není třeba zaplatit součet pokut za všechny opožděné programy, ale jen nejvyšší pokutu ze všech. Vašek se proto nyní snaží naplánovat práci na programech tak, aby zaplatil co možná nejnižší pokutu. Stejně jako dříve má i nyní k dispozici jen jeden počítač, a proto není možné pracovat na více programech najednou. Započatou práci na programu není možné přerušit.*

Soutěžní úloha: Na vstupu je pro každý z nedokončených programů uveden vzorec na výpočet pokuty a údaj, kolik dní práce je zapotřebí na jeho dokončení. Napište program, který určí rozvrh na dokončení programů, při němž Vašek zaplatí nejmenší pokutu. Vzorec na výpočet pokuty má tvar polynomu nejvýše třetího stupně $ax^3 + bx^2 + cx + d$, ve kterém jsou koeficienty a, b, c, d celočíselné nezáporné a x je počet dní, o něž se odevzdání programu opozdilo.

Vstup: První řádek vstupního souboru `attosoft.in` obsahuje kladné celé číslo N ($1 \leq N \leq 5\,000$) – počet programů. Následuje N řádků, i-tý z nich obsahuje pět celých čísel l_i, a_i, b_i, c_i, d_i ($1 \leq l_i \leq 100$, $0 \leq a_i, b_i, c_i, d_i \leq 5\,000$) kde l_i je počet dní potřebný na dokončení i-tého programu a a_i, b_i, c_i, d_i jsou koeficienty vzorce na výpočet pokuty. Můžete předpokládat, že za 100 000 dní se stihnou napsat všechny programy.

Výstup: Výstupní soubor `attosoft.out` obsahuje N čísel oddělených mezerami nebo konci řádků. Tato čísla představují čísla jednotlivých programů v pořadí, v němž je třeba programy dokončit, aby byla pokuta nejmenší možná. Pokud má úloha více řešení, vypište jedno libovolné z nich.

Příklad:

Vstupní soubor `attosoft.in`

```
3
10 1 0 0 0
3 0 0 0 10
1 0 0 5 0
```

Výstupní soubor `attosoft.out`

```
1
3
2
```

Zde pokuta za program číslo 1 dokončený po deseti dnech je $10^3 = 1000$, za program číslo 2 dokončený po 14 dnech je pokuta 10 a za program číslo 3 dokončený po 11 dnech je $5 \cdot 11 = 55$. Vašek tedy zaplatí pokutu 1000.

* Šikovnější z vás si po přečtení zbytku zadání uvědomí, že i kdyby se to smělo, stejně by se to nevyplatilo.