Toto sú návodné úlohy k domácemu kolu 37. ročníka Olympiády v informatike. Ide teda o sadu ľahších úloh, ktoré tematicky súvisia so súťažnými úlohami. Riešenie týchto úloh môže byť dobrou prípravou na riešenie súťažných úloh. Za riešenia týchto návodných úloh nie sú žiadne body do súťaže.

## Návodné úlohy k domácemu kolu OI: kategória A

### A-I-1 Tancujúci kráľ

1. Máme pole obsahujúce $n$ navzájom rôznych prvkov. Môžeme medzi sebou ľubovoľne vymieňať dvojice **susedných** prvkov. Dokážte, že pole vieme ľubovoľne preusporiadať – teda vytvoriť ľubovoľnú z $n!$ permutácií jeho prvkov.

2. Máme pole obsahujúce $n$ prvkov. Môžeme reverznúť ľubovoľný úsek poľa. Ukážte, že vieme medzi sebou vymeniť ľubovoľné dva prvky poľa (pričom obsah zvyšku poľa musí skončiť späť na svojich miestach).

   Vieme to vždy dosiahnuť pomocou nanajvýš piatich reverzov? Na najmenej koľko reverzov sa to zaručene dá spraviť?

3. Máme pole obsahujúce $n$ prvkov. Chceme ho usporiadať. Môžeme ho meniť len tak, že reverzneme ľubovoľný úsek poľa. (Máme teda jednoduchšiu verziu súťažnej úlohy: medzi tanečníkmi nie je kráľ, takže môžeme reverznúť úplne ľubovoľný úsek.)

   Vysvetlite, ako túto úlohu vyriešiť tak, že skombinujeme operáciu z predchádzajúcej návodnej úlohy a vhodne zvolený štandardný algoritmus na usporiadanie poľa.

   Viete touto technikou ľubovoľné pole usporiadať použitím $O(n \log n)$ reverzov?

4. Algoritmus z riešenia predchádzajúcej úlohy nie je optimálny, pole vieme usporiadať aj pomocou ešte menšieho počtu reverzov. Nájdite čo najefektívnejší algoritmus, ktorý to spraví.

   (Pomôcka: Existuje riešenie tejto úlohy, ktoré je myšlienkovo aj implementačne jednoduchšie ako riešenie predchádzajúcej úlohy. Zamyslite sa nad ňou teda aj vtedy, ak vám predchádzajúce dve úlohy robili problémy.)

5. Výzva pre súťažiacich s ambíciou reprezentovať Slovensko na IOI:

   Naďalej uvažujme tú istú úlohu: k danému poľu nájsť postupnosť reverzov, ktorá ho usporiada.

   Doteraz nás zaujímal len počet reverzov. Lenže čas, ktorý na ich nájdenie naše doterajšie riešenia potrebovali, bol rádovo väčší ako výsledný počet reverzov. Napríklad si všimnite, že priamočiara implementácia reverzu vyžaduje čas priamo úmerný dĺžke reverzovaného úseku (a teda vo všeobecnosti lineárny od dĺžky poľa).

   Nájdite riešenie tejto úlohy s čo najlepšou asymptotickou časovou zložitosťou. Dobrý program by mal byť schopný načítať pole dĺžky 200 000 a nanajvýš do pár sekúnd vygenerovať postupnosť reverzov, ktorá toto pole usporiada. Výstup by teda mal byť v rovnakom formáte ako v súťažnej úlohe: pre každý reverz index prvého a posledného políčka v reverzovanom úseku.

### A-I-2 Prominencia

1. Pole $H$ popisuje výšky bodov na hrebeni pohoria ako v zadaní súťažnej úlohy. Pripomíname, že vrcholy sú lokálne maximá hrebeňa: tie jeho body a vodorovné úseky, z ktorých ide hrebeň na obe strany dodola. Vrcholy druhého typu budeme volať náhorné plošiny.

   Napíšte program s lineárnou časovou zložitosťou, ktorý zistí, koľko má zadané pohorie vrcholov a koľko z nich je náhorných plošín.

2. Upravte predchádzajúci program tak, aby aj zistil, koľko vrcholov v danom pohorí je najvyšších na svete. (Viaceré vrcholy môžu mať tú istú výšku.)

3. Viete to celé spraviť na jeden prechod vstupom? (Takémuto programu by teoreticky malo stačiť len konštantne veľa premenných, pričom pole výšok raz postupne prečíta zo vstupu ale neukladá ho celé do pamäte.)

4. Každý by chcel mať najväčšiu horu sveta rovno pri svojej dedine, lebo je to dobré pre turistický ruch. A tak ľudia z okolia občas donesú do hôr kamene a svoju horu trochu navýšia.

Napíšte program, ktorý načíta pole $H$ popisujúce výšky bodov na hrebeni pohoria ako v zadaní súťažnej úlohy. Váš program potom musí čo najefektívnejšie spracúvať operácie tvaru „hodnota $H[i]$ stúpla o $x$“. (Meniť sa môžu všetky prvky poľa $H$. Tým môžu ľubovoľne vznikať a zanikať vrcholy.)

Po každej operácii by mal váš program oznámiť jeden index, na ktorom sa momentálne nachádza najvyšší vrchol sveta.

5. Upravte program z predchádzajúcej návodnej úlohy tak, aby vedel odpovedať aj na všeobecnejšie otázky nasledujúceho tvaru: „kde je momentálne najvyššia hora v úseku $H[i..j]$?“

## A-I-3 Strelec

1. Máme tabuľku rozmerov $6 \times 6$. Môžeme sa hýbať po jej políčkach, v každom kroku môžeme z aktuálneho políčka prejsť na iné, ktoré s ním susedí stranou. Koľkými rôznymi spôsobmi sa dá najkratšou cestou prejsť z ľavého horného rohu do pravého dolného?

2. Zovšeobecnite riešenie predchádzajúcej úlohy pre tabuľku $r \times s$.

   (Nájdite buď rozumne efektívny algoritmus, ktorý odpoveď vypočíta, alebo matematický vzorec pre odpoveď. Ak ste našli vzorec, zdôvodnite, prečo funguje.)

3. Uvažujme nasledujúcu tabuľku:

   ```
   ABRAKA
   BRAK-D
   RAKADA
   A-ADAB
   KADABR
   ADABRA
   ```

   Naďalej platí, že sa z aktuálneho políčka môžeme hýbať len na políčko s ním susediace stranou.

   Koľkými spôsobmi sa dá prejsť z ľavého horného rohu do pravého dolného tak, aby sme cestou prečítali slovo `ABRAKADABRA`? (Nesmieme teda stúpiť ani na jednu z pomlčiek.)

4. Navrhnite efektívny algoritmus riešiaci všeobecnú verziu vyššie uvedenej úlohy.

   (Na vstupe je tabuľka ľubovoľných písmen a slovo. Úlohou je spočítať cesty zľava hora doprava dole počas ktorých si prečítame práve zadané slovo.)

5. Vyriešte súťažnú úlohu pre šachového kráľa namiesto šachového strelca.

## A-I-4 Pokazený rover

1. V študijnom texte definujeme makro `pridaj X Y`, ktoré do lokality Y pridá toľko kamienkov, koľko je v lokalite X. Definujte iné podobné makro, ktoré bude robiť tradičnejšiu verziu priradenia.

   (Na konci behu makra teda má byť lokalita X nezmenená a v lokalite Y má byť rovnako kamienkov ako v lokalite X, bez ohľadu na to, koľko kamienkov bolo v lokalite Y na začiatku behu makra.)

2. Napíšte makro, ktoré (pomocou prázdnej pomocnej lokality) vymení obsah lokalít X a Y.

3. Napíšte program, ktorý spraví nasledovnú operáciu: ak je v lokalite X aspoň sedem kamienkov, jeden z nich odstráni (a inak nechá počet kamienkov nezmenený).

4. Napíšte program, ktorý spraví nasledovnú operáciu: ak je v lokalite X nanajvýš sedem kamienkov, jeden tam pridá (a inak nechá počet kamienkov nezmenený).

5. Napíšte program, ktorý spraví nasledovnú operáciu: ak je v lokalite X nanajvýš sedem kamienkov, doplní počet kamienkov na presne sedem (a inak nechá počet kamienkov nezmenený).

6. Napíšte program, ktorý zistí, či je počet kamienkov v lokalite X deliteľný siedmimi. Obsah lokality X smie váš program ľubovoľne zmeniť (na konci nemusí ostať zachovaný).

7. Napíšte program, ktorý bude počítať umocňovanie. Teda program, ktorý keď spustíme s tým, že v lokalite X je uložených $x$ kamienkov a v lokalite Y je $y$ kamienkov, tak do lokality Z uloží $x^y$ kamienkov.

   (Pre ľubovoľné $x$, vrátane $x = 0$, platí $x^0 = 1$.)