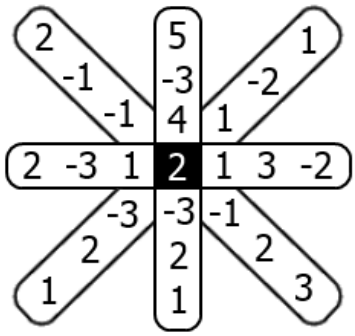
Piramida (piramida)

Piramida je popularna igra koja potiče iz drevnog Visokog. Ime je dobila po tome što tabla na kojoj se igra izgleda poput piramide gledane odozgo. Svaka takva tabla sačinjena je od *n* krakova, od kojih svaki krak ima tačno *k* brojeva. Svaki igrač ove igre ima na raspolaganju *p* poteza i potrebno je da dobije što više poena. U sredini table, tj. na vrhu piramide se nalazi specijalni broj *s*.



Primjer table sa četiri kraka sačinjena od šest brojeva

Glavno pravilo igre je da igrač može izabrati bilo koji krak od datih *n* krakova i zatim zaokružiti jedan dio jedne polovine kraka. Broj poena dobijen tim potezom je jednak zbiru brojeva koji se nalaze u zaokruženom dijelu jedne polovine kraka sabrano sa zbirom istim i simetričnim dijelom druge polovine kraka. Ukoliko bismo na primjeru sa slike izabrali brojeve 2 i 1 sa donje polovine vertikalnog kraka, dobili bismo ukupno 5 poena (1 + 2 + (-3) + 5 = 5). Nemoguće je izabrati više dijelova jednog kraka, tj. ne smije biti razmaka između jedne cjeline jedne polovine kraka, npr. izabrati -3 i 1 sa donjeg dijela vertikalnog kraka. Također, dozvoljeno je samo jednom zaokruživati brojeve sa jednog kraka. Igrač nastavlja igrati dok ne iskoristi sve poteze, iako bi neki potezi u najboljem slučaju donosili negativne poene. Specijalno pravilo važi za broj *s*. Tokom igre, igrač može iskoristiti pravilo da svaki broj jednak broju *s* u toku samo jednog poteza postane duplo veći. Ukoliko bi za datu tablu na gornjoj slici bilo iskorišteno to pravilo onda bi svi brojevi 2 u toku jednog poteza postali 4.

Zadatak

Vaš zadatak je da napišete funkciju *MaxUcinak* koja kao argumente prima cijele brojeve *n*, *k*, *p* i *s*, kao i niz cijelih brojeva (veličine $k \cdot n$) gdje redom svakih *k* brojeva predstavlja poene koji se nalaze na jednom od *n* krakova. Broj *k* je uvijek paran, broj *p* može imati vrijednosti od 1 do 100,

uključujući, dok je broj *s* pozitivan i manji ili jednak od 10. Na krakovima se mogu nalaziti poeni u vrijednosti od -10 do 10, uključujući. Redoslijed brojeva u nizu je takav da vertikalni krak dolazi prvi, a potom svi ostali krakovi u smjeru kazaljke na satu, s poenima od jednog kraja kraka prema drugom. Funkcija kao rezultat vraća cijeli broj koji predstavlja maksimalan broj poena koje je moguće osvojiti u *p* poteza, pritom poštujući prethodno opisana pravila igre.

Podzadatak 1 (12 bodova): $1 \leq n \leq 1.000$, $2 \leq k \leq 100$

Podzadatak 2 (13 bodova): $1 \leq n \leq 1.700$, $2 \leq k \leq 400$

Podzadatak 3 (14 bodova): $1 \leq n \leq 2.000$, $2 \leq k \leq 500$

Podzadatak 4 (61 bod): $1 \leq n \leq 8.000$, $2 \leq k \leq 1.000$

Primjeri

Primjer 1 (sa slike)

MaxUcinak (4, 6, 3, 2, {5, -3, 4, -3, 2, 1, 1, -2, 1, -3, 2, 1, -2, 3, 1, 1, -3, 2, 3, 2, -1, -1, -1, 2}) = 18

Obrazloženje: U prvom potezu se može iskoristiti specijalni broj *s*, koji je ovdje broj 2, i tako povećati sve brojeve koji su 2 na 4. Prvo ćemo zaokružiti brojeve 3 i 2 (pa tako i 2 i -1) na četvrtom kraku, što iznosi 10 poena. Poslije tog poteza, ploča se vraća na normalno stanje. Zatim zaokružimo broj 5 (pa tako i broj 1) na prvom kraku (vertikalni krak) što iznosi 6 poena. Zadnji potez ćemo iskoristiti na zaokruživanje broja 1 (pa tako i broja 1 na drugom kraju kraka) na drugom kraku i tako dobiti 2 poena. Zadnji potez bi se mogao odigrati i na zaokruživanje unutrašnje jedinice trećeg kraka (horizontalni krak) i tako dobiti 2 poena. Rezultat je 10 + 6 + 2 = 18 poena.

Primjer 2

MaxUcinak (2, 4, 2, 3, {7, -5, 9, 1, -4, 3, -5, 4}) = 13

Detalji implementacije

Sa servera za takmičenje možete preuzeti pripremljena okruženja (*piramida_c.zip*, *piramida_cpp.zip* ili *piramida_pas.zip*) sa osnovnim fajlovima za *C*, *C++* i *Pascal*.

Ukoliko koristite *C* ili *C++*, napišite funkciju s prototipom:
int MaxUcinak (**int** n, **int** k, **int** p, **int** s, **int*** poeni);

u fajlu *piramida.[c/cpp]*.

Ukoliko koristite *Pascal*, napišite funkciju sa prototipom:
function MaxUcinak (n : **LongInt**; k : **LongInt**; p : **LongInt**; s : **LongInt**; var poeni : **Array of LongInt**) : **LongInt**;
u fajlu *piramida.pas*.

Samo unutar ovog fajla treba da implementirate svoje rješenje. Pri tome smijete koristiti i druge pomoćne funkcije koje ste vi napisali, te standardna zaglavlja/biblioteke odabranog programskog jezika i funkcije iz ovih biblioteka. Ne smijete ni na koji način vršiti interakciju sa standardnim ulazom/izlazom niti sa bilo kojom datotekom.

U pripremljenom okruženju nalazi se fajl *grader.[c/cpp/pas]* koji testira ispravnost rada[1] funkcije koju ste napisali na javne testne primjere. Kada šaljete svoje rješenje, šaljete samo fajl *piramida.[c/cpp/pas]*, dok komisija koristi svoj *grader.[c/cpp/pas]* koji nije javni. U skladu s tim, slobodni ste da modificirate *grader.[c/cpp/pas]* i prilagođavate ga svojim potrebama u svrhu testiranja na lokalnom računaru.

Ukoliko koristite *Code::Blocks* u pripremljenim okruženjima možete naći i odgovarajuće projekte sa podešenim parametrima za prevođenje. *Release build* u potpunosti odgovara parametrima za prevođenje koji su na serveru za takmičenje, dok *Debug build* ima isključene optimizacije i uključene simbole za debagiranje.

Ukoliko koristite *FreePascal IDE*, dovoljno je da pokrenete prevođenje fajla *grader.pas* dok je u istom folderu fajl *piramida.pas*. Na serveru za takmičenje postavljeni su sljedeći parametri za prevođenje: -dEVAL -vw -XS -O2.

Ukoliko ne koristite *Code::Blocks*, odnosno *FreePascal IDE*, u okruženjima se nalaze i fajlovi *prevedi_[c/cpp/pas].sh* koje možete koristiti za prevođenje svojih programa, a koje pozivate iz terminala komandom sh *prevedi_[c/cpp/pas].sh* iz odgovarajućeg foldera.

1 Fajl grader.[c/cpp/pas] koji je javno dostupan testira samo ispravnost bez postavljanja ograničenja na vrijeme izvršavanja i iskorištenu memoriju.