Zadatke, test primjere i rješenja pripremili: Alen Rakipović, Frane Kurtović, Ivan Mandura, Ivo Sluganović, Tomislav Gudlek, Gustav Matula, Goran Gašić, Goran Žužić i Ante Đerek. Primjeri implementiranih rješenja su dani u priloženim izvornim kodovima koji nužno ne odgovaraju u svim detaljima ovdje opisanim algoritmima.

KAMEN

Predložio: Ante Đerek
Potrebno znanje: simulacija, dvodimenzionalna polja, while petlja

Zadatak se rješava direktnom implementacijom postupka padanja kamena koja je opisana u tekstu zadatka. U dvodimenzionalno polje znakova učitamo početno stanje ploče te ga tijekom cijelog programa koristimo kako bi opisali trenutno stanje ploče. Pomoću for petlje jedan po jedan kamen stavljamo u svaku ploču a za svaki kamen, unutar while petlje direktno implementiramo logiku iz teksta zadatka te ga pomičemo dolje po ploči sve dok možemo.

TIPKE

Predložio: Ante Đerek
Potrebno znanje: polja, geometrija, Pitagorin poučak (euklidska udaljenost točaka)

Prvi dio zadatka traži da se za svaku točku odredi kojoj tipki pripada. Osnovni podatak iz kojega se može vrlo jednostavno izračunati pripada li točka određenoj tipki su koordinate središta tipke, te računanje tih vrijednosti predstavlja glavni dio ovog problema. Jedno od jednostavnijih rješenja je obilaziti tipke po redcima tipkovnice, te unutar reda računati središta povećavajući X koordinatu za 320 (razmak između središta dviju susjednih tipaka u istom retku). Prelazak između redaka se računa smanjujući Y koordinatu za 220. Sada se za točku odredi pripadna tipka tako da se pronađe najbliže središte i provjeri je li točka unutar pravokutnika koji je određen koordinatama središta, te širinom i visinom koje su zadane u zadatku. Ta provjera je vrlo jednostavna i radi se tako da se izračuna apsolutna vrijednost razlike X koordinata središta i točke te ta vrijednost mora biti manja od pola širine. Isto se napravi i za Y koordinatu. Točka je unutar pravokutnika samo ako oba dva uvjeta vrijede. Pametne implementacije će raspored tipaka na tipkovnici spremiti u konstantu tipa string, te prolaziti kroz njih pomoću for petlje što je brže i otpornije na greške nego ručna isprobavati jedno po jedno slovo pomoću puno if naredbi.

Drugi dio zadatka se rješava prolaskom po svim riječima iz rječnika, računanjem njihove udaljenosti i ispisivanje one s najmanjom udaljenošću. Udaljenost riječi od niza točaka se računa za svako slovo zasebno, tako da se zbroje udaljenosti odgovarajuće točke i odgovarajućeg slova. U prošlom dijelu zadatka su već izračunata središta pravokutnika, tako da su svi podaci dostupni i potrebno je samo provesti opisani postupak.

ASCII

Predložio: Goran Žužić
Potrebno znanje: polja, površina, obilazak grafa (DFS ili BFS)

U zadatku je potrebno pronaći površinu poligona sastavljenog od znakova '/' i '\' koji u sebi sadrži proizvoljni broj pregrada koje trebamo zanemariti. Problem je pritom određivanje koje su pregrade na rubu, a koje unutar samog poligona.

Tome je najlakše pristupiti pomoću širenja s ruba slike prema rubu poligona pomoću algoritma obilaska grafa (BFS ili DFS). Širimo se sa svake prazne pozicije u 8 susjednih pozicija dok ne naiđemo na rub samog poligona. Nakon što smo identificirali svaki rub, konačna površina se može izračunati po formuli broj_kvadratića_koji_čine_rub/2 + broj_kvadratića_unutar_poligona.

Za one koji žele više

Ako se unutar poligona ne smiju nalaziti pregrade (kao u prvih 50% test primjera), zadatak je moguće riješiti na jednostavniji način. Primijetimo da se pojedini prazni kvadratić nalazi unutar poligona ako i samo ako se u njegovom retku lijevo od njega nalazi neparan broj prepreka. Koristeći ovu činjenicu moguće je jednostavno pronaći broj praznih kvadratića koji su unutar poligona.

Za one koji žele manje

Sljedeći C program sadrži točno 100 znakova ne računajući razmak i znak prelazak u novi red, ispravno se prevodi sa gcc kompajlerom te osvaja 50% bodova na ovom zadatku. Napravi još kraći program koji dobiva barem 50% bodova.

```
f(x,c){c=getchar();return c<0?0:c-46?(c-47)%45?f(x)
:1+f(x^1):f(x)+x*2;} main(){printf("%d\n",f(0)/2);}
```

PROZORI

Predložio: Ivo Sluganović
Potrebno znanje: dvodimenzionalna polja, osnovne petlje

Zbog malih ograničenja bilo je dovoljno za svaki prozor isprobavati sve moguće pozicije dvjema for-petljama, te provjeravati je li pozicija prazna još dvjema (ugniježđenim) for-petljama. Nakon što pronađemo dobru poziciju dvjema petljama crtamo zadani oblik u matricu.

TETRIS

Predložio: Goran Žužić
Potrebno znanje: dvodimenzionalna polja, osnovne petlje, pohlepni algoritam

Svi su barem jednom u životu igrali tetris i upoznati su s jednostavnim skupom pravila ove klasične igre. Međutim, ovdje je problem pronaći način kako uz pomoć vertikalnog bloka "I" (4 spojena kvadratića u ravnoj liniji) počistiti cijelu ploču osim eventualno najdonja 3 retka.

Najjednostavnije rješenje zadatka se dobije ako primijetite kako je uvijek dovoljno pustiti blok u stupac u kojem može najdublje (najniže) propasti. Nije se teško na papiru uvjeriti kako će dobiveni algoritam uvijek dovoditi do točnog rješenja. Potom je potrebno samo pažljivo implementirati pravila same igre kako bi se dobili svi bodovi.

DODIR

Predložio: Ante Đerek
Potrebno znanje: stringovi, dinamičko programiranje, Pitagorin poučak (euklidska udaljenost točaka)

Rješenje prvog dijela zadatka je opisano u zadatku Tipke, a najbitniji dio postupka je računanje koordinata središta svake tipke što se koristi i u rješavanju drugog podzadatka. U drugom podzadatku se traži ispisivanje riječi koja je najbliža zadanom nizu točaka. Potrebno je koristeći tri vrste operacija pretvoriti niz točaka tako da se i-ta točka nalazi unutar pravokutnika i-tog slova riječi. Naravno, cilj je odrediti najmanju moguću cijenu transformacija. Ovaj problem se može efikasno riješiti dinamičkim programiranjem. Neka A i B redom označavaju niz znakova (ciljnu riječ) i početni niz točaka.

Funkciju f(a,b) definiramo kao najmanju moguću cijenu niza transformacija koji će pretvoriti sufiks originalnog niza točaka B koji počinje indeksom b da odgovara sufiksu niza znakova A koji počinje indeksom a.

$$f(a,b) = min\{f(a+1,b) + CA, f(a,b+1) + CD, f(a+1,b+1) + d(A[a],B[b])\}$$

Ova formula, tj. prijelaz isprobava sve tri transformacije, te uzima najjeftiniju kao rješenje. Prvi član pokušava dodati novu točku koja će odgovarati slovu A[a] uz cijenu CA. Drugi član briše b-tu točku uz cijenu CB. Treći član pomiče točku B[b] u središte tipke A[a] uz cijenu koju računa funkcija d.

Ova funkcija je rekurzivna, te je još potrebno definirati uvjete zaustavljanja. Kada parametri a ili b dosegnu kraj niza tada je rješenje trivijalno. Ako parametar a prvi dosegne kraj niza slova tada je rješenje pobrisati sve preostale točke, a ako parametar b prvi dosegne kraj niza točaka tada je rješenje dodati onoliko točaka koliko je slova preostalo u nizu slova.

Kod implementiranja ove rekurzije je potrebno u vanjskom polju pamtiti izračunate rezultate funkcije, kako ne bi došlo do višekratnog izračunavanja iste funkcije, taj postupak se zove memoizacija.

Vremenska i memorijska složenost računanja ove funkcije je O(N*K), gdje je K duljina riječi, te se taj postupak ponavlja za svaku riječ u rječniku.

Za one koji žele više

Razmislite kako biste ovaj algoritam implementirali bez korištenja rekurzije (iterativno). Ideja je obilaziti stanja na način da je trenutno stanje moguće izračunati iz već izračunatih vrijednosti funkcije (samo je potrebno na pametan način postaviti granice for petlje kojom se obilaze stanja).

Nakon što to riješite, probajte iskoristiti tu metodu da smanjite potrošnju memorije, tj. da ne pamtite sva stanja.