1. tekmovanje IJS v znanju računalništva za srednješolce

6. maja 2006

PRAVILA TEKMOVANJA ZA TRETJO SKUPINO

Vsaka naloga zahteva, da napišeš program, ki prebere neke vhodne podatke, izračuna odgovor oz. rezultat ter ga izpiše v izhodno datoteko. Programi naj berejo vhodne podatke iz datoteke *imenaloge*.`in` in izpisujejo svoje rezultate v *imenaloge*.`out`. Natančni imeni datotek sta podani pri opisu vsake naloge. V vhodni datoteki je vedno po en sam testni primer. Vaše programe bomo pognali po večkrat, vsakič na drugem testnem primeru. Besedilo vsake naloge natančno določa obliko (format) vhodnih in izhodnih datotek. Tvoji programi lahko predpostavijo, da se naši testni primeri ujemajo s pravili za obliko vhodnih datotek, ti pa moraš zagotoviti, da se bo izpis tvojega programa ujemal s pravili za obliko izhodnih datotek.

Delovno okolje

Na začetku boš dobil mapo s svojim uporabniškim imenom ter navodili, ki jih pravkar prebiraš. Ko boš sedel pred računalnik, boš dobil nadaljnja navodila za prijavo v sistem.

Na vsakem računalniku imaš na voljo enoto (disk) `U:`, na kateri lahko kreiraš svoje datoteke (datoteke, ki so tam že od prej, pusti pri miru). Programi naj bodo napisani v programskem jeziku Pascal, C, C++ ali Java, mi pa jih bomo preverili z 32-bitnimi prevajalniki FreePascal, GNU C/C++ in GCJ. Za delo lahko uporabiš `FP` oz. `ppc386` (FreePascal), `GCC`/`G++` (GNU C/C++ — command line compiler), `GCJ`, `GPC` in Java 2 SDK.

Oglej si tudi spletno stran: `http://rtk/`, kjer boš dobil nekaj testnih primerov in program `RTK.EXE`, ki ga lahko uporabiš za preverjanje svojih rešitev. Tukaj si lahko tudi ogledaš anonimizirane rezultate ostalih tekmovalcev.

Preden boš oddal prvo rešitev, boš moral programu za preverjanje nalog sporočiti svoje ime, kar bi na primer Janez Novak storil z ukazom

```
rtk -name JNovak
```

(prva črka imena in priimek, brez presledka, brez šumnikov).

Za oddajo rešitve uporabi enega od naslednjih ukazov:

`rtk` *imenaloge*`.pas`
`rtk` *imenaloge*`.c`
`rtk` *imenaloge*`.cpp`
`rtk` *ImeNaloge*`.java`

Program `rtk` bo prenesel izvorno kodo tvojega programa na testni računalnik, kjer se bo prevedla in pognala na desetih testnih primerih. Na spletni strani boš dobil za vsak testni primer obvestilo o tem, ali je program pri njem odgovoril pravilno ali ne. Če se bo tvoj program s kakšnim testnim primerom ukvarjal več kot deset sekund, ga bomo prekinili in to šteli kot napačen odgovor pri tem testnem primeru.

Da se zmanjša možnost zapletov pri prevajanju, ti priporočamo, da ne spreminjaš privzetih nastavitev svojega prevajalnika. Tvoji programi naj uporabljajo le standardne knjižnice svojega programskega jezika in naj ne delajo z datotekami na disku, razen s predpisano vhodno in izhodno datoteko. Dovoljena je uporaba literature (papirnate), ne pa računalniško berljivih pripomočkov (razen tega, kar je že na voljo na tekmovalnem računalniku), prenosnih računalnikov, prenosnih telefonov itd.

Ocenjevanje

Vsaka naloga lahko prinese tekmovalcu od 0 do 100 točk. Vsak oddani program se preizkusi na desetih testnih primerih; pri vsakem od njih dobi od 0 do 10 točk (praviloma 10, če je izpisal popolnoma pravilen odgovor, sicer pa 0; izjemi sta 1. in 5. naloga, kjer dobijo boljše rešitve več točk kot slabše), nato pa se te točke po vseh testnih primerih seštejejo v skupno število točk tega programa. Če si oddal N programov za to nalogo in je najboljši med njimi dobil M (od 100) točk, dobiš pri tej nalogi $\max\{0, M-3(N-1)\}$ točk. Z drugimi besedami: za vsako oddajo (razen prve) pri tej nalogi se ti odbijejo tri točke. Pri tem pa ti nobena naloga ne more prinesti negativnega števila točk. Če nisi pri nalogi oddal nobenega programa, ti ne prinese nobenih točk. Če se poslana izvorna koda ne prevede uspešno, to ne šteje kot oddaja.

Skupno število točk tekmovalca je vsota po vseh nalogah. Tekmovalce razvrstimo po skupnem številu točk.

Vsak tekmovalec se mora sam zase odločiti o tem, katerim nalogam bo posvetil svoj čas, v kakšnem vrstnem redu jih bo reševal in podobno. Verjetno je priporočljivo najprej reševati lažje naloge.

Poskusna naloga (ne šteje k tekmovanju) (`poskus.in`, `poskus.out`)

Napiši program, ki iz vhodne datoteke prebere eno celo število (le-to je v prvi vrstici, okoli njega ni nobenih dodatnih presledkov ipd.) in izpiše njegov desetkratnik v izhodno datoteko.

Primer vhodne datoteke:

```
123
```

Ustrezna izhodna datoteka:

```
1230
```

Primer rešitve:

```
program PoskusnaNaloga;
var T: text; i: integer;
begin
  Assign(T, 'poskus.in'); Reset(T); ReadLn(T, i); Close(T);
  Assign(T, 'poskus.out'); Rewrite(T); WriteLn(T, 10 * i); Close(T);
end.
```

```
#include <stdio.h>
int main() {
  FILE *f = fopen("poskus.in", "rt");
  int i; fscanf(f, "%d", &i); fclose(f);
  f = fopen("poskus.out", "wt"); fprintf(f, "%d\n", 10 * i);
  fclose(f); return 0;
}
```

```
#include <fstream>
int main() {
  std::ifstream ifs("poskus.in"); int i; ifs >> i;
  std::ofstream ofs("poskus.out"); ofs << 10 * i;
  return 0;
}
```

```
import java.io.*;
public class Poskus {
  public static void main (String[] args) {
    try {
      StreamTokenizer st = new StreamTokenizer(new FileReader("poskus.in"));
      st.nextToken(); int i = (int) st.nval;
      PrintWriter os = new PrintWriter(new FileOutputStream("poskus.out"));
      os.println(10 * i); os.close();
    } catch (Exception e) { }
  }
}
```

1. tekmovanje IJS v znanju računalništva za srednješolce

6. maja 2006

NALOGE ZA TRETJO SKUPINO

Rešitve bodo objavljene na `http://rtk.ijs.si/`.

1. Optična miška (`miska.in`, `miska.out`)

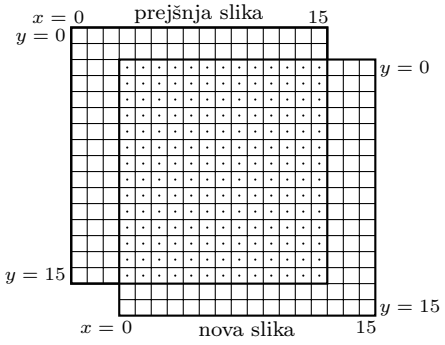
Optične računalniške miške, ki ne potrebujejo posebne podlage, vsebujejo preprosto, a hitro videokamero, ki si stalno ogleduje del podlage pod miško, vgrajeni procesor pa zaporedne slike primerja med seboj. Glede na to, da uporabnik premika miško dokaj počasi in neskokovito glede na kratek čas med zaporednima slikama (pod tisočinko sekunde), sta si zaporedni sliki precej podobni, naslednja slika je kvečjemu nekoliko premaknjena glede na prejšnjo (sukanje lahko zanemarimo, dviga miške s podlage pri tej nalogi ne bomo predvideli).

Vsaka slika je sestavljena iz kvadratne mreže $n \times n$ slikovnih elementov (pikslov). Ker zajema kamera le sivinske slike, je vsak slikovni element predstavljen s svetlostjo, ki je celo število med 0 in 63. Točka s koordinatama $(0, 0)$ je v zgornjem levem vogalu slike. Predpostavimo, da so premiki miške med dvema zaporednima slikama veliki kvečjemu tri piksle v vsaki smeri ($-3 \leq dx \leq 3$, $-3 \leq dy \leq 3$).

S primerjanjem dveh zaporednih slik bi radi ocenili, kakšen je najverjetnejši premik miške v času med trenutkoma, ko sta bili sliki posneti.

Nek možni premik (dx, dy) ocenimo takole: za vsak slikovni element, ki bi bil pri tem premiku viden tako na prejšnji kot na trenutni sliki, izračunamo absolutno vrednost razlike v svetlosti te točke na prejšnji in na trenutni sliki. Nato izračunamo povprečje teh absolutnih vrednosti po vseh slikovnih elementih, ki so vidni tako na prejšnji kot na trenutni sliki. Manjše ko je to povprečje, bolj verjetno je, da je (dx, dy) res pravi premik.

Na primer: če delamo s slikami velikosti 16×16 in se miška premakne za $dx = 3$, $dy = 2$, je prejšnji in trenutni sliki skupnih $13 \cdot 14$ slikovnih elementov in po njih bi računali povprečje iz gornjega odstavka. Na spodnji ilustraciji so ti skupni slikovni elementi označeni s pikami.



Napiši program, ki bo prebral nekaj parov zaporednih slik iz vhodne datoteke in za vsak par izpisal najverjetnejši premik miške v horizontalni in vertikalni smeri.

Vhodna datoteka: v prvi vrstici je celo število m ($0 < m \leq 30$), ki pove, da v tem testnem primeru nastopa m parov slik. Sledijo podatki o posameznih parih slik, pred vsakim od njih pa je prazna vrstica.

Za vsak par slik je v vhodni datoteki najprej vrstica, ki vsebuje celo število n ($4 \leq n \leq 64$), ki pove, da sta sliki v tem paru veliki $n \times n$ slikovnih elementov. Sledi prazna vrstica, nato pa n vrstic, ki opisujejo sliko pred premikom (od zgoraj navzdol: prva

vrstica je za $y = 0$, zadnja za $y = n - 1$); v vsaki od teh vrstic je n celih števil (med vključno 0 in vključno 63), ki povedo svetlost slikovnih elementov te vrstice slike (od leve proti desni: prvo število je za $x = 0$, zadnje za $x = n - 1$). Nato pride še ena prazna vrstica in nato še nadaljnjih n vrstic, ki na enak način podajajo drugo sliko (tisto po premiku miške).

Izhodna datoteka: za vsak par slik poišči zamik (dx, dy), pri katerem je dosežena najmanjša vrednost zgoraj opisane ocene (povprečna razlika absolutnih vrednosti razlik svetlosti tistih slikovnih elementov, ki so skupni stari in novi sliki). Če je ta najmanjša vrednost dosežena pri več zamikih, je vseeno, katerega izpišeš. Veljavni zamiki so le tisti, pri katerih sta dx in dy celi števili in velja $-3 \leq dx \leq 3$, $-3 \leq dy \leq 3$. Najdeni zamik za vsak par slik izpiši v svojo vrstico (najprej dx, nato dy, vmes pa presledek) in to v enakem vrstnem redu, v kakršnem so pari slik podani v vhodni datoteki. Vmes ne izpisuj praznih vrstic ali česa podobnega.

Točkovanje: če izhodna datoteka ni oblikovana tako, kot je opisano v prejšnjem odstavku, dobi tvoj program pri tem testnem primeru 0 točk. Drugače pa, če je od m premikov pravilno prepoznal k premikov, dobi pri tem testnem primeru $(10 \cdot k)$ div m točk.

Primer vhodne datoteke:

```
2

4

56 52 47 40
57 48 43 39
56 50 44 36
56 50 41 35

53 50 45 40
56 54 49 46
52 46 42 34
51 45 38 32

5

53 42 33 26 17
45 36 30 23 13
37 30 23 20 12
27 25 19 15 12
19 16 12 11 9

36 45 43 36 28
35 46 36 31 23
37 47 27 24 18
38 47 19 16 13
39 45 9 9 9
```

Pripadajoča izhodna datoteka:

```
1 -2
-2 1
```

Pri prvem paru slik lahko na primer vidimo, da se slikovni elementi

```
52 47 40
48 43 39
```

na stari sliki

precej lepo ujemajo z elementi

```
52 46 42
51 45 38
```

na novi sliki.

Podobno se pri drugem paru lepo ujemajo

```
45 36 30
37 30 23
27 25 19
19 16 12
```

na stari in

```
43 36 28
36 31 23
27 24 18
19 16 13
```

na novi sliki.

2. Spletne knjigarne (`knjigarne.in`, `knjigarne.out`)

Prek spletnih knjigarn bi radi kupili po en izvod vsake izmed n knjig. Na voljo imamo k spletnih knjigarn; posamezno knjigo lahko kupimo od katerekoli od njih, vendar pa so lahko cene knjig v različnih knjigarnah različne. Plačati moramo tudi dostavo knjig na naš domači naslov. Knjigarna i ($1 \leq i \leq k$) računa a_i denarnih enot za dostavo prve naročene knjige in b_i za dostavo vsake naslednje naročene knjige (če iz neke knjigarne ne naročimo nobene knjige, ji seveda ni treba plačati ničesar). Vedno velja $a_i \geq b_i$. Cena knjige j ($1 \leq j \leq n$) pri knjigarni i je c_{ji} denarnih enot.

Skupna cena, ki jo bomo morali plačati za želenih n knjig, je torej v splošnem lahko odvisna od tega, pri kateri knjigarni bomo naročili katero knjigo. **Napiši program**, ki izračuna najmanjšo skupno ceno knjig (s stroški dostave vred), za katero lahko nakupimo vseh n knjig.

Vhodna datoteka: v prvi vrstici sta celi števili n in k, ločeni s presledkom. Zanju velja $1 \leq n \leq 10000$ in $1 \leq k \leq 10$. Sledi še $n + 2$ vrstic; v vsaki je po k celih števil, ločenih s po enim presledkom. V prvi od teh vrstic so cene $a_1, a_2, \ldots, a_k$; v drugi so cene $b_1, b_2, \ldots, b_k$; v ostalih vrsticah pa so cene posamezne knjige v vseh k knjigarnah: najprej je tu vrstica s cenami $c_{11}, c_{12}, \ldots, c_{1k}$ (cena prve knjige v vseh k knjigarnah), nato vrstica s cenami $c_{21}, c_{22}, \ldots, c_{2k}$ (cena prve knjige v vseh k knjigarnah) in tako naprej. Zadnja vrstica torej vsebuje cene $c_{n1}, c_{n2}, \ldots, c_{nk}$. Vse cene, a_i, b_i, c_{ji} za vse i od 1 do k in vse j od 1 do n, so cela števila, večja ali enaka 1 in manjša ali enaka 10000.

Izhodna datoteka: vanjo izpiši celo število, ki je najnižja skupna cena (s stroški dostave vred), za katero je mogoče dobiti vseh n knjig.

Primer vhodne datoteke:

```
10 3
8 10 10
5 7 10
10 8 11
10 9 12
9 9 12
8 7 11
9 9 12
9 9 14
9 9 10
9 10 11
8 8 11
10 8 3
```

Pripadajoča izhodna datoteka:

```
142
```

Za ta denar lahko pridemo do knjig, če kupimo vse razen zadnje knjige v prvi knjigarni, zadnjo pa v tretji.

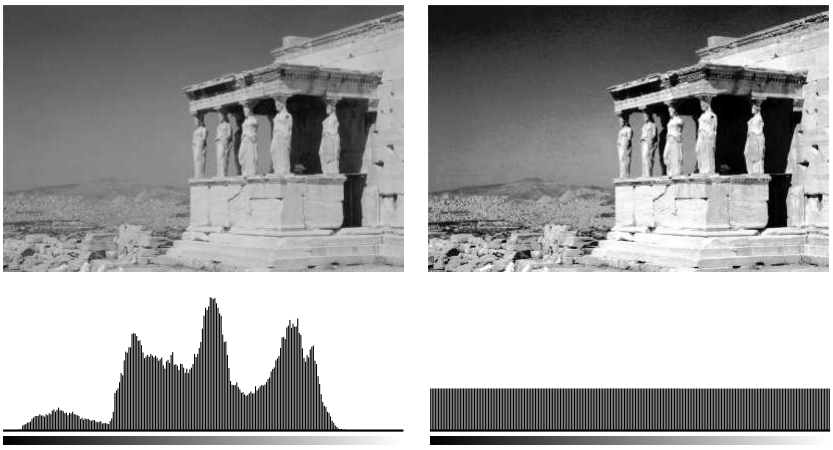
3. Izravnavanje histogramov (`histogrami.in`, `histogrami.out`)

Imamo sivinsko sliko, široko w slikovnih elementov (pikslov) in visoko h slikovnih elementov. Barva posameznega slikovnega elementa je opisana z enim od celih števil od 0 do 255 (večja ko je, svetlejši je ta slikovni element — 0 pomeni črno, 255 pa belo barvo).

Če za vsako svetlost od 0 do 255 preštejemo, koliko slikovnih elementov na sliki ima ravno to svetlost, in ta števila predstavimo s stolpci, dobimo *histogram* naše slike.

Izravnavanje histograma (*histogram equalization*) je postopek, pri katerem sliko popravimo tako, da postane njen histogram čim bolj „raven“ — torej da so vsi stolpci približno enako visoki.

Primer slike pred in po izravnavanju histograma (pod vsako različico slike je narisan tudi histogram):



Napiši program, ki prebere vhodno sliko I in izpiše izhodno sliko I' (enake velikosti kot I), za katero velja:

- Število slikovnih elementov najpogostejše barve na sliki I' je kvečjemu za 1 večje od števila slikovnih elementov najredkejše barve (tiste, ki ji pripada najmanj slikovnih elementov).
- Za vsak par slikovnih elementov (x_1, y_1) in (x_2, y_2) velja: če je slikovni element (x_1, y_1) na sliki I svetlejši od slikovnega elementa (x_2, y_2), je na sliki I' slikovni element (x_1, y_i) svetlejši ali pa enake barve kot (x_2, y_2), ni pa temnejši.

Če obstaja več slik I', ki ustrezajo gornjima pogojema, je vseeno, katero izpišeš.

Vhodna datoteka: v prvi vrstici sta celi števili h (višina slike) in w (širina slike), ločeni s presledkom. Obe sta večji ali enaki 1 in manjši ali enaki 200. Sledi h vrstic, v vsaki od njih pa je w celih števil, ločenih s po enim presledkom; i-to število v j-ti vrstici pove barvo slikovnega elementa na preseku i-tega stolpca in j-te vrstice slike. Vse barve so večje ali enake 0 in manjše ali enake 255.

Izhodna datoteka: vanjo zapiši h vrstic, v vsaki od njih pa naj bo w celih števil, ločenih s po enim presledkom. Ta števila naj opisujejo barve slikovnih elementov izhodne slike na enak način, kot je opisana vhodna slika v vhodni datoteki. Izhodna slika mora ustrezati zahtevam, navedenim v besedilu naloge.

Primer vhodne datoteke:

```
5 5
105 105 99 101 101
104 102 103 104 103
105 9 103 105 103
9 105 105 101 102
102 9 103 101 101
```

Ena od možnih pripadajočih izhodnih datotek:

```
19 20 3 4 5
17 9 12 18 13
21 0 14 22 15
1 23 24 6 10
11 2 16 7 8
```

4. Mafija (`mafija.in`, `mafija.out`)

V neki mafijski družini so člani urejeni hierarhično: vsakdo razen vrhovnega šefa ima natanko enega nadrejenega človeka in če sledimo povezavam od podrejenega do nadrejenega, lahko od vsakega člana družine sčasoma pridemo do vrhovnega šefa.

Denar v družini kroži takole. Tisti, ki nimajo nobenih podrejenih, morajo zbirati denar z raznimi kriminalnimi dejavnostmi in ves tako prisluženi denar oddati svojemu nadrejenemu. Ta mora ves denar, ki ga prejme od podrejenih, oddati *svojemu* nadrejenemu, on ga pošlje spet svojemu nadrejenemu in tako naprej, vse dokler ves denar ne konča v rokah vrhovnega šefa.

Vendar pa šef sumi, da nekateri člani družine goljufajo in ne pošljejo naprej svojemu nadrejenemu vsega denarja, ki so ga prejeli od svojih podrejenih. Šef je z izsiljevanjem in vohunjenjem uspel od vsakega člana pridobiti podatek o tem, koliko denarja je poslal svojemu nadrejenemu, zdaj pa prosi tebe, da mu **napišeš program**, ki bo ugotovil, kolikšna je največja vsota denarja, ki jo je utajil kakšen od članov družine (torej največja razlika med tem, koliko denarja je prejel od podrejenih, in tem, koliko ga je posredoval svojemu nadrejenemu). Za tiste, ki nimajo nobenega podrejenega, predpostavimo, da niso utajili nič svojega zaslužka (ker ne znamo oceniti, koliko denarja so v resnici zbrali s svojimi kriminalnimi dejavnostmi).

Vhodna datoteka: v prvi vrstici je celo število n ($0 < n \leq 100000$), ki pove, koliko članov ima družina. Člani družine so oštevilčeni s celimi števili od 1 do n. Sledi še n vrstic datoteke, ki za vsakega člana družine povedo, kdo je njegov nadrejeni in koliko denarja je ta član posredoval temu svojemu nadrejenemu. Tako torej $(i+1)$-va vrstica datoteke vsebuje dve celi števili, ločeni s presledkom; prvo od teh števil je številka člana, ki je neposredno nadrejen članu i, drugo pa je znesek denarja, ki ga je i oddal svojemu nadrejenemu (ta znesek je večji ali enak 0, obenem pa manjši ali enak vsoti zneskov, ki jih je član i prejel od svojih podrejenih). Pri vrhovnem šefu, ki nadrejenega nima, sta navedeni dve ničli. Vsota zneskov, ki jih posamezni član družine prejme od svojih podrejenih, pri nobenem članu ne presega 10^9 denarnih enot.

Izhodna datoteka: za vsakega člana (razen vrhovnega šefa) izračunaj razliko med zneskom, ki ga je prejel od podrejenih, in zneskom, ki ga je oddal nadrejenemu. Izpiši največjo vrednost te razlike po vseh članih.

Primer vhodne datoteke:

```
8
3 100
3 50
4 120
0 0
8 30
8 40
8 50
4 95
```

Pripadajoča izhodna datoteka:

```
30
```

Član 3 je od svojih podrejenih (1 in 2) prejel 150 denarnih enot, oddal pa 120, torej jih je utajil 30. Član 8 pa je od svojih podrejenih (5, 6 in 7) prejel 120 denarnih enot, oddal pa jih je 95, torej jih je utajil 25.

5. Tovornjaki (`tovornjaki.in`, `tovornjaki.out`)

Gneče na trajektih so običajen pojav in Jadrolinija vlaga veliko naporov v to, da bi jih zmanjšala. Običajno na obali stoji kolona tovornjakov in potrebno je čimprej ugotoviti, kako velik trajekt potrebujemo, da jih lahko naekrat prepeljemo.

Dano imamo torej zaporedje tovornjakov, ki bi jih radi razporedili na trajekt. Na trajektu bodo tovornjaki stali v treh pasovih in mi si lahko poljubno izberemo, na katerem pasu bo stal kateri tovornjak. Seveda pa skupna dolžina tovornjakov na nobenem od pasov ne sme presegati dolžine trajekta. **Napiši program**, ki bo za dano zaporedje dolžin tovornjakov poskusil ugotoviti, kako dolg je najkrajši trajekt, pri katerem bi se še dalo razporediti te tovornjake v tri pasove, ne da bi skupna dolžina tovornjakov na katerem od pasov presegla dolžino trajekta.

Vhodna datoteka: v prvi vrstici je celo število n ($1 \leq n \leq 100$), ki pove število tovornjakov. Sledi še n vrstic, v katerih so dolžine tovornjakov. Vsaka dolžina tovornjaka je celo število, večje ali enako 1 in manjše ali enako 100.

Izhodna datoteka: V prvo vrstico izpiši dolžino najkrajšega trajekta, na katerega je tvojemu programu uspelo razporediti tovornjake iz vhodne datoteke v skladu z zahtevami naloge. (Testni primeri bodo zasnovani tako, da ta dolžina najkrajšega trajekta nikoli ne bo večja od 100.) Sledi naj še n vrstic, ki opišejo nek konkreten razpored tovornjakov na ta najkrajši trajekt: v i-ti izmed teh vrstic naj bo število 1, 2 ali 3, ki pove, na kateri pas je bil razporejen i-ti tovornjak iz vhodne datoteke.

Točkovanje: če tvoja izhodna datoteka ne ustreza zgoraj opisanim zahtevam, dobiš pri tistem testnem primeru 0 točk, drugače pa je število točk odvisno od tega, kako dolg je tvoj trajekt v primerjavi z najkrajšim možnim. Recimo, da je pri nekem testnem primeru najkrajši možni trajekt dolg t^* enot, v tvoji izhodni datoteki pa je trajekt dolžine t enot. Potem dobiš pri tem testnem primeru $\max\{1, 10 - (t - t^*)\}$ točk. Z drugimi besedami, 10 točk dobiš le, če je tvoja rešitev najboljša možna ($t = t^*$); sicer pa se ti odbije toliko točk, za kolikor je tvoj trajekt daljši od najkrajšega možnega, vendar največ devet točk (tako da zagotovo dobiš vsaj eno točko).

Primer vhodne datoteke:

```
7
26
15
29
30
30
16
10
```

Pripadajoča izhodna datoteka:

```
55
1
2
1
2
3
3
2
```

Na trajekt dolžine 55 lahko razporedimo tovornjake takole: $26 + 29$ na en pas, $10 + 15 + 30$ na drugega in $16 + 30$ na tretjega. To je pri tem zaporedju tovornjakov tudi najkrajši možni trajekt.