1. tekmovanje ACM v znanju računalništva za osnovne šole

Šolsko tekmovanje

26. januarja 2024

NASVETI ZA TEKMOVALCE

Naloge na tem šolskem tekmovanju pokrivajo širok razpon težavnosti, tako da ni nič hudega, če ne znaš rešiti vseh.

Psevdokodi pravijo včasih tudi strukturirani naravni jezik. Postopek opišemo v naravnem jeziku, vendar opis strukturiramo na podoben način kot pri programskih jezikih, tako da se jasno vidi strukturo vejitev, zank in drugih programskih elementov.

Primer opisa postopka v psevdokodi: recimo, da imamo zaporedje besed in bi ga radi razbili na več vrstic tako, da ne bo nobena vrstica preširoka.

```
naj bo trenutna vrstica prazen niz;
pregleduj besede po vrsti od prve do zadnje:
    če bi trenutna vrstica z dodano trenutno besedo (in presledkom
    pred njo) postala predolga,
        izpiši trenutno vrstico in jo potem postavi na prazen niz;
    dodaj trenutno besedo na konec trenutne vrstice;
če trenutna vrstica ni prazen niz, jo izpiši;
```

(Opomba: samo zato, ker je tu primer psevdokode, to še ne pomeni, da moraš tudi ti pisati svoje odgovore v psevdokodi.)

Če pa v okviru neke rešitve pišeš izvorno kodo programa ali podprograma, obvezno poleg te izvorne kode v nekaj stavkih opiši, kako deluje (oz. naj bi delovala) tvoja rešitev in na kakšni ideji temelji.

Pri ocenjevanju so vse naloge vredne enako število točk. Svoje odgovore dobro utemelji. Prizadevaj si predvsem, da bi bile tvoje rešitve pravilne, ob tem pa je zaželeno, da so tudi čim bolj učinkovite (take dobijo več točk kot manj učinkovite). Za manjše sintaktične napake se načeloma ne odbije veliko točk. Priporočljivo in zaželeno je, da so tvoje rešitve napisane pregledno in čitljivo. Če je na listih, ki jih oddajaš, več različic rešitve za kakšno nalogo, jasno označi, katera je tista, ki naj jo ocenjevalci upoštevajo.

Če naloga zahteva branje ali obdelavo vhodnih podatkov, lahko tvoja rešitev (če v nalogi ni drugače napisano) predpostavi, da v vhodnih podatkih ni napak (torej da je njihova vsebina in oblika skladna s tem, kar piše v nalogi).

Če oblika vhodnih in izhodnih podatkov ni natančno določena, lahko tvoj program obravnava bere in piše podatke v poljubni obliki.

Nekatere naloge zahtevajo branje podatkov s standardnega vhoda in pisanje na standardni izhod. Za pomoč je tu nekaj primerov programov, ki delajo s standardnim vhodom in izhodom:

- Program, ki prebere s standardnega vhoda dve števili in izpiše na standardni izhod njuno vsoto:

```
program BranjeStevil;
var i, j: integer;
begin
  ReadLn(i, j);
  WriteLn(i, ' + ', j, ' = ', i + j);
end. {BranjeStevil}
```

```
#include <stdio.h>
int main() {
  int i, j; scanf("%d %d", &i, &j);
  printf("%d + %d = %d\n", i, j, i + j);
  return 0;
}
```

- Program, ki bere s standardnega vhoda po vrsticah, jih šteje in prepisuje na standardni izhod, na koncu pa izpiše še skupno dolžino:

```
program BranjeVrstic;
var s: string; i, d: integer;
begin
  i := 0; d := 0;
  while not Eof do begin
    ReadLn(s);
    i := i + 1; d := d + Length(s);
    WriteLn(i, '. vrstica: "', s, '"');
  end; {while}
  WriteLn(i, ' vrstic, ', d, ' znakov.');
end. {BranjeVrstic}
```

```
#include <stdio.h>
#include <string.h>
int main() {
  char s[201]; int i = 0, d = 0;
  while (gets(s)) {
    i++; d += strlen(s);
    printf("%d. vrstica: \"%s\"\n", i, s);
  }
  printf("%d vrstic, %d znakov.\n", i, d);
  return 0;
}
```

Opomba: C-jevska različica gornjega programa predpostavlja, da ni nobena vrstica vhodnega besedila daljša od dvesto znakov. Funkciji gets se je v praksi bolje izogibati, ker pri njej nimamo zaščite pred primeri, ko je vrstica daljša od naše tabele s. Namesto gets bi bilo bolje (in varneje) uporabiti fgets ali fscanf; vendar pa za rešitev naših tekmovalnih nalog zadošča tudi gets.

- Program, ki bere s standardnega vhoda po znakih, jih prepisuje na standardni izhod, na koncu pa izpiše če število prebranih znakov (ne vštevši znakov za konec vrstice):

```
program BranjeZnakov;
var i: integer; c: char;
begin
  i := 0;
  while not Eof do begin
    while not Eoln do
      begin Read(c); Write(c); i := i + 1 end;
    if not Eof then begin ReadLn; WriteLn end;
  end; {while}
  WriteLn('Skupaj ', i, ' znakov.');
end. {BranjeZnakov}
```

```
#include <stdio.h>

int main() {
  int i = 0, c;
  while ((c = getchar()) != EOF) {
    putchar(c); if (i != '\n') i++;
  }
  printf(Škupaj %d znakov.\n", i);
  return 0;
}
```

Še isti trije primeri v pythonu:

```
# Branje dveh števil in izpis vsote:
import sys

a, b = sys.stdin.readline().split()
a = int(a); b = int(b)
print "%d + %d = %d"% (a, b, a + b)
```

```
# Branje standardnega vhoda po vrsticah:
import sys

i = d = 0
for s in sys.stdin:
  s = s.rstrip('\n') # odrežemo znak za konec vrstice
  i += 1; d += len(s)
  print "%d. vrstica: \"%s\""% (i, s)
print "%d vrstic, %d znakov."% (i, d)
```

```
# Branje standardnega vhoda znak po znak:
import sys

i = 0
while True:
  c = sys.stdin.read(1)
  if c == : break # EOF
  sys.stdout.write(c)
  if c != '\n': i += 1
print Škupaj %d znakov."% i
```

Še isti trije primeri v javi:

```
// Branje dveh števil in izpis vsote:
import java.io.*;
import java.util.Scanner;

public class Primer1
{
   public static void main(String[] args) throws IOException
   {
      Scanner fi = new Scanner(System.in);
      int i = fi.nextInt(); int j = fi.nextInt();
      System.out.println(i + "+ "+ j + "= "+ (i + j));
   }
}
```

```
// Branje standardnega vhoda po vrsticah:
import java.io.*;

public class Primer2
{
   public static void main(String[] args) throws IOException
   {
      BufferedReader fi = new BufferedReader(new InputStreamReader(System.in));
      int i = 0, d = 0; String s;
      while ((s = fi.readLine()) != null) {
         i++; d += s.length();
         System.out.println(i + ". vrstica: \""+ s + "\""); }
      System.out.println(i + "vrstic, "+ d + žnakov.");
   }
}
```

```
// Branje standardnega vhoda znak po znak:
import java.io.*;

public class Primer3
{
   public static void main(String[] args) throws IOException
   {
      InputStreamReader fi = new InputStreamReader(System.in);
      int i = 0, c;
      while ((c = fi.read()) >= 0) {
         System.out.print((char) c); if (c != '\n' && c != '\r') i++; }
      System.out.println(Škupaj "+ i + žnakov.");
   }
}
```

1. tekmovanje ACM v znanju računalništva za osnovne šole

Šolsko tekmovanje

26. januarja 2024

NALOGE ZA ŠOLSKO TEKMOVANJE

Svoje odgovore dobro utemelji. Če pišeš izvorno kodo programa ali podprograma, **OBVEZNO** tudi v nekaj stavkih z besedami opiši idejo, na kateri temelji tvoja rešitev. če ni v nalogi drugače napisano, lahko tvoje rešitve predpostavljajo, da so vhodni podatki brez napak (da ustrezajo formatu in omejitvam, kot jih podaja naloga). Zaželeno je, da so tvoje rešitve poleg tega, da so pravilne, tudi učinkovite (bolj učinkovite rešitve dobijo več točk). Naloge so štiri in pri vsaki nalogi lahko dobiš od 0 do 25 točk.

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

1. Podaje

Otroci stojijo v krogu in si podajajo žogo. Vsak otrok ujame podajo svojega levega soseda in jo poda naprej svojemu desnemu sosedu, prva je žogo podala Maja. Učiteljica jim je naročila, naj vsak prešteje kolikokrat je žogo podal naprej, hkrati pa naj šteje tudi skupno število vseh podaj. Maja je preštela, da so vsi skupaj naredili k podaj, pozabila pa je šteti svoje podaje. Od pouka matematike se spomni, da je to mogoče izračunati, vendar se ne spomni kako. Pomagaj ji.

Napiši program (ali podprogram, ali psevdokodo), ki kot vhod dobi podano število otrok v krogu in število podaj in izračuna, kolikokrat je žogo podala Maja.

Oglejmo si dva primera:

Vhod:	Pripadajoči izhod:
`5 10`	`2`

V tem primeru je žoga naredila točno dva kroga. Maja je podala žogo v prvi in šesti podaji. Po deseti podaji je žogo spet ujela, vendar je ni več podala naprej.

Vhod:	Pripadajoči izhod:
`8 20`	`3`

V tem primeru je žoga naredila dva kroga in pol. Maja je žogo podala v prvi, deveti in sedemnajsti podaji.

2. Pribor

Klemen zlaga žlice iz pomivalnega stroja v predal s priborom. Žlice so različnih velikosti, iz stroja pa jih jemlje po vrsti v točno določenem vrstnem redu. V predalu jih zlaga na kupe. Klemen želi vse žlice zložiti na čim manj kupov, zato začne z enim samim kupom. Vendar pa večjih žlic ne more zlagati na manjše žlice, zato mora vsakič, ko iz stroja vzame žlico, ki je večja od prejšnje, končati s trenutnim kupom in začeti zlagati na nov kup. če sta dve zaporedni žlici enako veliki, mu novega kupa ni treba začeti. Zdaj bi rad izračunal, kako veliki bodo njegovi kupi.

Napiši program (ali podprogram, ali psevdokodo), ki prebere število n - število žlic v Majinem pomivalnem stroju in potem še n števil, ki predstavljajo velikosti njenih žlic. Za vsak kup zloženih žlic izpiši eno število - koliko žlic je Klemen zložil na ta kup.

Vhod:

```
4
1
2
3
4
```

Pripadajoči izhod:

```
1
1
1
1
```

Vhod:

```
4
4
3
2
1
```

Pripadajoči izhod:

```
4
```

Vhod:

```
6
1
3
3
2
6
5
```

Pripadajoči izhod:

```
1
3
2
```

3. Palindromski oklepaji

Palindrom je besedilo, ki se enako bere z leve in z desne, kot npr. beseda *neradodaren*. Oklepajski izraz `()()` ni palindrom, ker ima bran iz desne obliko `)()(`. Izraz pa ima kljub temu posebno obliko: enak je svoji zrcalni sliki.

Napiši program (ali podprogram, ali psevdokodo), ki bo določil, če je besedilo, sestavljeno iz oklepajev `(`, `)`, `[`, `]`, palindrom, in če je morda enako svoji zrcalni sliki.

Podrobneje si oglejmo nekaj primerov:

Vhod:	Pripadajoči izhod:
`()()`	`zrcalen izraz`

V tem primeru je prvi znak zrcalen četrtemu in drugi tretjemu, kar pomeni, da je izraz zrcalen.

Vhod:	Pripadajoči izhod:
`][][`	`palindrom`

V tem izrazu je prvi znak enak petemu, drugi pa četrtemu. Izraz je torej palindrom.

Vhod:	Pripadajoči izhod:
`([]`	`navaden`

Ta izraz ni ne zrcalen ne palindromski.

4. Slika iz kock

Filipova mlajša sestra je sestavila pravokotnik iz raznobarvnih kock tako, da sestavljene kocke narišejo sliko. Filip je trenutno v šoli v naravi in slike ne more videti, zato mu je sestra barve kock sporočila po telefonu. Barve je opisala tako, da je najprej podala zaporedno številko vrstice in stolpca, v kateri je kocka, in nato njeno barvo. Ker ni želela, da Filip prehitro vidi, kaj je na sliki, je opise barv kock podajala v naključnem vrstnem redu. Žal pa se je pri štetju uštela, in je pozabila sporočiti barvo nekaterih kock.

Napiši program (ali podprogram ali psevdokodo), ki sprejme sestrin opis slike, in izračuna velikost slike ter položaje kock, na katere je Filipova sestra pozabila. Predpostaviš lahko, da ima sestra največ 5000 kock, in da se zaporedne številke vrstic in stolpcev začnejo pri 1 in ne presegajo 5000. Poleg tega je sestra zagotovo sporočila barvo neke kocke v zadnjem stolpcu, in neke kocke v zadnji vrstici.

Barve na sliki so označene s števili med ena in miljon. Opis slike je sestavljen iz števila n, ki mu sledi še n vrstic s po tremi števili, x_i, y_i in b_i. Prvi dve števili predstavljata položaj kocke v koordinatni mreži, zadnje pa njeno barvo.

Vhod:	Pripadajoči izhod:
`5`	
`1 1 5`	`3 3`
`2 3 4`	`2 1`
`1 2 4`	`3 3`
`3 1 2`	`3 2`
`2 2 1`	`1 3`

V danem primeru je slika dimenzije 3×3. Preostale vrstice izhoda predstavljajo manjkajoče kocke. Vrstni red izpisanih manjkajočih kock ni pomemben.

1. tekmovanje ACM v znanju računalništva za osnovne šole

Šolsko tekmovanje

26. januarja 2024

REŠITVE NALOG ŠOLSKEGA TEKMOVANJA

1. Podaje

Maja žogo poda enkrat na začetku vsakega kroga. Število zaključenih krogov dobimo s celoštevilskim deljenjem k/n. Vendar pa je iz drugega primera v nalogi razvidno, da to ni vedno pravi odgovor. Lahko se namreč zgodi, da po vseh zaključenih krogih še kakšna podaja ostane. Tedaj bo Maja žogo podala še enkrat in količniku moramo prišteti ena. Za izračun končnega odgovora moramo torej preveriti, ali je število podaj deljivo s številom otrok v krogu. To lahko naredimo na primer takole v jeziku C++:

```
#include <stdio.h>

int main() {
  int n, k;
  scanf("%d%d", &n, &k);
  int odgovor = k / n;
  if (k % n != 0) {
    odgovor += 1;
  }
  printf("%d\n", odgovor);
}
```

Ali pa v Pythonu:

```
l = input().split()
n = int(l[0])
k = int(l[1])

odgovor = k // n
if k % n != 0:
  odgovor += 1
print(odgovor)
```

V Pythonu moramo biti pozorni, da uporabimo operator `//` namesto `/`, saj z `/` ne bi dobili celoštevilskega rezultata.

Za preverjanje deljivosti smo uporabili operator `%`, ki izračuna ostanek pri celoštevilskem deljenju. Vemo, da ostanek nič pomeni, da se deljenje izide. Pogoj bi lahko napisali tudi kot `if (odgovor * n != k)`. Pri celoštevilskem deljenju namreč velja deljenec = delitelj * količnik + ostanek. Zgornji pogoj bo torej veljal natanko tedaj ko bo ostanek neničeln.

Obstaja pa še tretja možnost. Celoštevilsko deljenje si lahko predstavljamo tudi kot običajno deljenje, ki pa rezultat zaokroži na največje celo število, ki je manjše od rezultata običajnega deljenja, torej navzdol. V tej nalogi pa si želimo rezultat zaokrožiti navzgor. Z nekaj spretnosti je mogoče odgovor na to vprašanje izračunati brez pogojnega stavka: `odgovor = (k + n - 1) / n`

Ob zgornjem razmisleku o zaokroževanju se morda postavlja vprašanje, zakaj ne bi rezultata dejansko izračunali tako da vhodni števili delimo v tipu ‘float‘ ali ‘double‘ in dobljeni decimalni rezultat zaokrožimo navzgor. Žal pri računanju z decimalnimi števili v računalniku pride do zaokrožitvenih napak, ki bi lahko povzročile, da s tem postopkom ne bi dobili pravega rezultata.

2. Pribor

Naloga od nas zahteva, da v zaporedju števil iščemo strnjena padajoča podzaporedja, torej kose seznama, v katerih je vsak element manjši ali enak prejšnjemu. Uporabimo eno spremenljivko, s katero štejemo velikost trenutnega kupa. Med branjem vhodnih podatkov jo sproti povečujemo, ali pa jo izpišemo in ponastavimo ko se vzorec prekine. Primera rešitev:

```
int main() {
  int n, odgovor, trenutna, prejsnja;
  scanf("%d", &n):
  scanf("%d", &prejsnja);
  odgovor = 1;
  for (int i = 1; i < n; i++) {
    scanf("%d", &trenutna);
    if (trenutna <= prejsnja) {
      odgovor += 1;
    } else {
      printf("%d\n", odgovor);
      odgovor = 1;
    }
    prejsnja = trenutna;
  }
  return 0;
}
```

in

```
n = int(input())
prejsnja = int(input())
odgovor = 1
for i in range(1, n):
  trenutna = int(input())
  if trenutna <= prejsnja:
    odgovor += 1
  else :
    print(odgovor)
    odgovor = 1
  prejsnja = trenutna
```

Pozorni moramo biti, da prvo žlico obravnavamo posebej in primerjave s prejšnjo žlico opravljamo šele od druge naprej.

3. Palindromski oklepaji

Opazimo, da niz ne more biti hkrati palindrom in zrcalen. Če je palindrom, je zadnji znak namreč enak prvemu, in ni njegova zrcalna slika. Lastnosti lahko torej preverjamo posebej. Za vsak znak v besedilu moramo preveriti, če je istoležni znak na drugi strani niza enak, ali zrcalna slika. Če ena od teh lastnosti za nek znak ne velja, vemo, da niz ni palindrom oz. zrcalen. Preverjanje zrcalnosti zahteva kar nekaj tipkanja:

```
#include <stdio.h>
#include <string.h>

char s[1000000]; // predpostavimo da bo niz krajsi od 1000000

int main() {
  scanf("%s", s);
  int n = strlen(s);
  int p = 1, z = 1;
  for (int i = 0; i < n; i++) {
    if (s[i] != s[n-i-1]) {
      p = 0;
    }
```

```
    if (s[i] == '(' && s[n-i-1] != ')'
        || s[i] == ')' && s[n-i-1] != '('
        || s[i] == '[' && s[n-i-1] != ']'
        || s[i] == ']' && s[n-i-1] != '[') {
      z = 0;
    }
  }
  if (z == 1) {
    printf(žrcalen\n");
  } else if (p == 1) {
    printf("palindrom\n");
  } else {
    printf("navaden\n");
  }
  return 0;
}
```

V Pythonu se lahko kompliciranemu pogoju izognemo z uporabo slovarja:

```
s = input()
p = z = True

zrcalno = {
    '(': ')',
    ')': '(',
    '[': ']',
    ']': '[',
}

for i in range(len(s)):
  if s[i] != s[n-i-1]:
    p = False
  if s[i] != zrcalno[s[n-i-1]]:
    z = False

if p:
  print('palindrom')
elif z:
  print('zrcalen')
else :
  print('navaden')
```

4. Slika iz kock

Ker zagotovo vemo, da se neka kocka na opisu nahaja v zadnjem stolpcu in neka v zadnji vrstici, lahko izračunamo višino in širino slike, ki sta natanko največja številka vrstice in največja številka stolpca. Ker vemo, da je vseh kock kvečjemu 5000 vemo, da se bo celotna slika zagotovo prilegala v tabelo velikosti 5000 * 5000 (ekstremna primera sta slika, široka eno kocko in visoka 5000 kock, ter slika, široka 5000 kock in visoka eno kocko). Zato lahko v tabeli brez skrbi da bi prekoračili meje zapisujemo, o katerih kockah smo podatke že prejeli. Barva posameznih kock nas pri tem sploh ne zanima. Ko prejmemo vse podatke se lahko samo še sprehodimo čez celotno tabelo in odčitavamo, katere informacije nam manjkajo. Primera rešitev:

```
#include <stdio.h>

int a[5001][5001];

int main() {
  int mx=0, my=0, n;
  scanf("%d", &n);
  for (int i = 0; i < n; i++) {
    int x, y, b;
```

```
    scanf("%d%d%d", &x, &y, &b);
    if (mx < x) {
      mx = x;
    }
    if (my < y) {
      my = y;
    }
    a[y][x] = 1;
  }
  printf("%d %d\n", mx, my);
  for (int y = 1; y <= my; y++) {
    for (int x = 1; x <= mx; x++) {
      if (a[y][x] == 0) {
        printf("%d %d\n", x, y);
      }
    }
  }
  return 0;
}
```

in

```
a = []
for i in range(5001):
  a.append([0]*5001)

mx = 0
my = 0
n = int(input())
for i in range(n):
  l = input().split()
  x = int(l[0])
  y = int(l[1])
  a[y][x] = 1
  mx = max(mx, x)
  my = max(my, y)

print(mx, my)

for y in range(1, my+1):
  for x in range(1, mx+1):
    if a[y][x] == 0:
      print(x, y)
```

1. tekmovanje ACM v znanju računalništva za osnovne šole

Šolsko tekmovanje

26. januarja 2024

NASVETI ZA MENTORJE O IZVEDBI TEKMOVANJA IN OCENJEVANJU

Tekmovalci naj pišejo svoje odgovore na papir ali pa jih natipkajo z računalnikom; ocenjevanje teh odgovorov poteka v vsakem primeru tako, da jih pregleda in oceni mentor (in ne npr. tako, da bi se poskušalo izvorno kodo, ki so jo tekmovalci napisali v svojih odgovorih, prevesti na računalniku in pognati na kakšnih testnih podatkih).

Glede tega, katere programske jezike tekmovalci uporabljajo, naše tekmovanje ne postavlja posebnih omejitev, niti pri nalogah, pri katerih je rešitev v nekaterih jezikih znatno krajša in enostavnejša kot v drugih (npr. uporaba perla ali pythona pri problemih na temo obdelave nizov).

Kjer se v tekmovalčevem odgovoru pojavlja izvorna koda, naj bo pri ocenjevanju poudarek predvsem na vsebinski pravilnosti, ne pa na sintaktični. Pri ocenjevanju na državnem tekmovanju zaradi manjkajočih podpičij in podobnih sintaktičnih napak odbijemo mogoče kvečjemu eno točko od petindvajsetih; glavno vprašanje pri izvorni kodi je, ali se v njej skriva pravilen postopek za rešitev problema. Ravno tako ni nič hudega, če npr. tekmovalec v rešitvi v C-ju pozabi na začetku **#include**ati kakšnega od standardnih headerjev, ki bi jih sicer njegov program potreboval; ali pa če podprogram `main()` napiše tako, da vrača **void** namesto **int**.

Pri vsaki nalogi je možno doseči od 0 do 25 točk. Od rešitve pričakujemo predvsem to, da je pravilna (= da predlagani postopek ali podprogram vrača pravilne rezultate), poleg tega pa je zaželeno tudi, da je učinkovita (manj učinkovite rešitve dobijo manj točk).

Če tekmovalec pri neki nalogi ni uspel sestaviti cele rešitve, pač pa je prehodil vsaj del poti do nje in so v njegovem odgovoru razvidne vsaj nekatere od idej, ki jih rešitev tiste naloge potrebuje, naj vendarle dobi delež točk, ki je približno v skladu s tem, kolikšen delež rešitve je našel.

Če v besedilu naloge ni drugače navedeno, lahko tekmovalčeva rešitev vedno predpostavi, da so vhodni podatki, s katerimi dela, podani v takšni obliki in v okviru takšnih omejitev, kot jih zagotavlja naloga. Tekmovalcem torej načeloma ni treba pisati rešitev, ki bi bile odporne na razne napake v vhodnih podatkih.

Če oblika vhodnih podatkov ni natančno določena, si lahko podrobnosti tekmovalec izbere sam. Na primer, če naloga pravi, da dobimo seznam parov, je to lahko v praksi tabela (*array*), vektor, *linked list* ali še kaj drugega, pari pa so lahko bodisi strukture, ki jih je deklarirala tekmovalčeva rešitev, ali pa kaj iz standardne knjižnice (kot je `pair` v C++ ali `tuple` v pythonu).

V nadaljevanju podajamo še nekaj nasvetov za ocenjevanje pri posameznih nalogah.

1. Podaje

- Rešitve, ki pravilno ugotovijo da je treba vhodni števili celoštevilsko deliti, vendar se pri tem ne ozirajo na deljivost - torej v nobenem primeru količniku ne prištejejo ena, ali pa v vseh primerih količniku prištejejo ena, naj dobijo kvečjemu 10 točk.

- Rešitve, ki uporabljajo napačno metodo omenjeno na koncu razlage uradne rešitve te naloge, naj dobijo kvečjemu 15 točk.

- Rešitve, ki nalogo rešujejo z zanko v kateri ponavljajo odštevanje n od k dokler k ne pade do nič, naj dobijo kvečjemu 11 točk, zaradi neučinkovitosti.

- Rešitvam, ki sicer ciljajo v pravo smer, vendar uporabljajo deljenje števil s plavajočo vejico namesto celoštevilskega deljenja (denimo zato, ker je tekmovalec zamešal `//` in `/` v Pythonu), naj se za to odbije 5 točk.
- Rešitvam, ki delajo pravilno razen za primere, ko je $k \leq n$, naj se za to odbije 5 točk.

2. Pribor

- Pri tej nalogi je ključni del to, da tekmovalci vpeljejo spremenljivko za prejšnjo vrednost in znajo trenutno vrednost primerjati s prejšnjo in posodobiti prejšnjo vrednost na koncu ponovitve. Rešitev, ki zadane te ključne elemente, naj dobi vsaj 15 točk.
- Rešitvam, ki namesto $<$ uporabljajo $<=$ ali pa namesto $>$ uporabljajo $>=$, naj se za to odbije 5 točk.
- Rešitvam, ki izvedejo napačno število primerjav s prešnjim (mora jih biti $n - 1$), pozabijo izpisati zadnji rezultat ali podobno, naj se za to odbijejo 4 točke.
- Rešitvam, ki v principu delujejo pravilno, vendar najprej celoten seznam shranijo nekam v spomin in potem berejo prejšnje elemente iz tega seznama namesto naravnost s standardnega vhoda, naj se za to odbije 2 točki zaradi neučinkovite rabe spomina.

3. Palindromski oklepaji

- Rešitve, ki naloge ne rešijo, naj dobijo delne točke, če so v njih razvidne opazke: 1. da niz ne more biti hkrati palindrom in zrcalen ali 2. da niz lihe dolžine ne more biti zrcalen. Za prvo opazko naj dobijo 6 točk in za drugo 3 točke.
- Rešitev, ki pravilno preverja palindromskost, naj dobi vsaj 9 točk.
- Rešitev, ki pravilno preverja zrcalnost, naj dobi vsaj 13 točk.

4. Slika iz kock

- Rešitev, ki pravilno izračuna dimenzije slike, naj za to dobi vsaj 8 točk.
- Rešitev, ki si vhodne podatke hrani v seznam, nato izračuna dimenzije tabele in šele nato rezervira ravno prav veliko tabelo, je ekvivalentna uradni rešitvi in naj se ji zaradi tega ne odbija točk.
- Rešitvi, ki ni pozorna na to, da se indeksi stolpcev in vrstic začnejo pri ena naj se odbije:
  - 6 točk, če je posledično rezervirana tabela premajhna in bi program zaradi tega lahko dostopal do spomina izven tabele
  - 3 točke, če je tabela kljub temu dovolj velika
- Rešitev, ki sicer ni pravilna, vendar pravilno opazi, da so vrednosti b_i nepomembne, naj za to dobi 2 točki.

Težavnost nalog

Državno tekmovanje ACM v znanju računalništva za osnovne šole poteka v dveh skupinah (prva je lažja, druga pa težja); na tem šolskem tekmovanju pa je skupina ena sama, vendar naloge v njej pokrivajo razmeroma širok razpon zahtevnosti. Za občutek povejmo, s katero skupino državnega tekmovanja so po svoji težavnosti primerljive posamezne naloge letošnjega šolskega tekmovanja:

Naloga	Kam bi sodila po težavnosti na državnem tekmovanju ACM
1. Podaje	lažja naloga v prvi skupini
2. Pribor	srednje težka naloga v prvi ali lažja v drugi skupini
3. Palindromski oklepaji	težka naloga v prvi ali srednja v drugi skupini
4. Slika iz kock	težka naloga v prvi ali srednja naloga v drugi skupini

Če torej na primer neki tekmovalec reši le eno ali dve lažji nalogi, pri ostalih pa ne naredi (skoraj) ničesar, to še ne pomeni, da ni primeren za udeležbo na državnem tekmovanju; pač pa je najbrž pametno, če gre na državnem tekmovanju v prvo skupino.