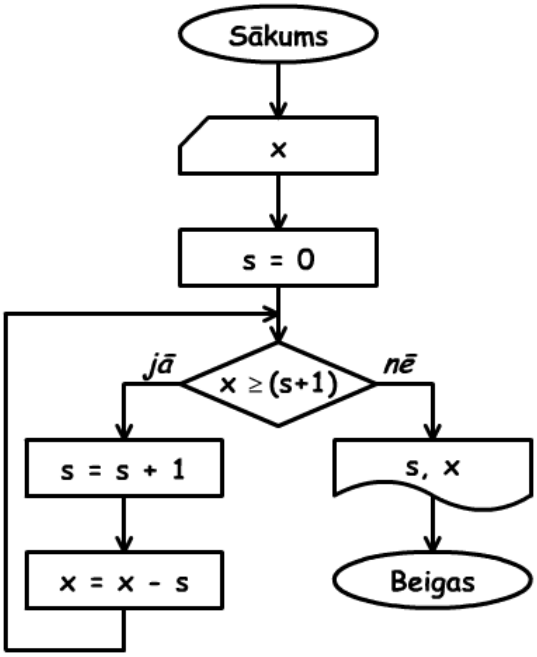
Liepājas Universitātes
Datorzinātņu olimpiādes 2014
studentu programmēšanas konkursa uzdevumi

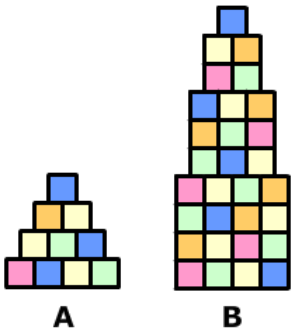
1. uzdevums [*uzd1*] (5 punkti)
Zināms, ka pēc dotās blokshēmas vai algoritma strukturētā apraksta var noteikt, cik stāvu torni var uzbūvēt un cik klucīšu paliks pāri, ja to būvē, kā attēlots zīmējumā A.



Sākums
1. **solis.** Izvēlas mainīgā *x* vērtību;
2. **solis.** Mainīgajam s piešķir sākuma vērtību 0;
3. **solis.** Ja *x* ≥ (s+1), tad pāriet uz 4. soli, citādi uz 6. soli;
4. **solis.** *s* vērtību palielina par 1;
5. **solis.** *x* vērtību samazina par *s*;
6. **solis.** Izvada *s* un *x* vērtības

Beigas

Sastādi programmu, kas ļautu noteikt, cik stāvu torni var uzbūvēt un cik klucīšu paliks pāri, ja to būvē, kā attēlots zīmējumā *B*.



2. uzdevums [*uzd2*] (5 punkti)
Uzzīmēt blokshēmu vai aprakstīt algoritmu strukturētā teksta veidā, pēc kura būtu iespējams noteikt, cik gadā dienu – 365 vai 366, ja lietotājs ievada gadu raksturojošu četrciparu skaitli (ļaut lietotājam ievadīt tikai četrciparu skaitli). Piebilde, gadā ir 366 dienas, ja gada skaitlis dalās ar 4, izņemot tos, kas dalās ar 100, bet nedalās ar 400.

3. uzdevums [*uzd3*] (10 punkti)
Sastādi programmu, kas ekrānā izvada *n* x *n* tabulas vērtības. Tabulas vērtības veidojas pēc attēlā redzamā principa. Lietotājs ievada n vērtību, n vērtība var būt tikai skaitlis intervālā [1;20].
Piemēram, ja n = 7, tad tabulas vērtības ir sekojošas:

1	**2**	**3**	**4**	**5**	**6**	**7**
2	**3**	**4**	**5**	**6**	**7**	**6**
3	**4**	**5**	**6**	**7**	**6**	**5**
4	**5**	**6**	**7**	**6**	**5**	**4**
5	**6**	**7**	**6**	**5**	**4**	**3**
6	**7**	**6**	**5**	**4**	**3**	**2**
7	**6**	**5**	**4**	**3**	**2**	**1**

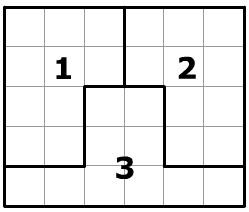
4. uzdevums [*uzd4*] (10 punkti)
Sastādi programmu, kas realizē doto algoritmu.

1. solis. Lietotājs ievada veselu skaitli *n*, intervālā no 10 līdz 20.
2. solis. Masīva A[*n*] vērtības aizpilda ar nejaušiem skaitļiem intervālā no 10 līdz 50.
3. solis. Ekrānā izvada masīva A[*n*] elementus.
4. solis. A[0] vērību saglabā mainīgajā *m1*.
5. solis. A[1] vērību saglabā mainīgajā *m2*.
6. solis. Mainīgajam *k* piešķir vērtību 2.
7. solis. Ja masīva A[*k*] elementa vērtība ir mazāka nekā *m1*, tad *m1* piešķir A[*k*] elementa vērtību.
8. solis. *k* vērtību palielina par 1.
9. solis. Ja masīva A[*k*] elementa vērtība ir lielāka nekā *m2*, tad *m2* piešķir A[*k*] elementa vērtību.
10. solis. *k* vērtību palielina par 1.
11. solis. Ja *k* vērtība ir mazāka par *n*, pāriet pie 7.soļa izpildes, citādi pāriet pie 12.soļa izpildes.
12. solis. Ekrānā izvada *m1* vērtību.
13. solis. Ekrānā izvada *m2* vērtību.
14. solis. Mainīgajam *m* piešķir mainīgo *m1* un *m2* summu.
15. solis. Ekrānā izvada *m* vērtību.

Ko dara (ko spēj aprēķināt) dotais algoritms?

5. uzdevums [*uzd5*] (15 punkti)
Uz planētas, kuru apdzīvo roboti, iedzīvotāji nemīl lietot decimālo skaitīšanas sistēmu, tādēļ, lai sazinātos ar viņiem, visi decimālie skaitļi ir jāaizstāj ar binārajiem skaitļiem. Piemēram, ja lietotāja ievadītais ziņojuma teksts ir, **HG2kj7##12NZ432p**, tad pārveidotais ziņojums ir **HG10kj111##1100NZ110110000p**.
Sastādīt programmu, kas lietotāja ievadīto ziņojumu – simbolu virkni, kuras garums nepārsniedz 250 simbolus, pārveido un izdrukā ekrāna robotiem pieņemamā veidā.

6. uzdevums [*uzd6*] (10 punkti)
Kārlis topošo arhitektu konkursam veido maketu. Tā kā materiāls, kuru viņš plāno izmantot maketa sienu izgatavošanai nav lēts, viņam iepriekš jāveic plānošanas un aprēķina darbi.
Kārlis uz rūtiņu papīra uzzīmējis ēkas starpsienu plānojumu. Zināms, ka vienas rūtiņas izmērs dzīvē būs 10 cm x 10 cm, bet sienu augstums būs 30 cm.



Sastādi programmu, kas palīdz Kārlim aprēķināt (izdrukā failā *uzd6.out*), cik kvadrātmetrus materiāla nepieciešams iegādāties, lai izgatavotu iecerēto maketu, ja faila *uzd6.in* pirmajā rindā doti divi veseli skaitļi *n* un *m* atdalīti ar atstarpi ($1 \le n, m \le 100$) - ēkas izmērs rūtiņās, bet katrā no sekojošajām faila *n* rindiņām doti *m* skaitļi atdalīti ar atstarpi, kas apzīmē katras rūtiņas piederību telpai plānā.

Piemēram,
ievaddati *uzd6.in*

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

izvaddati *uzd6.out*

```
1.02
```

7. uzdevums [*uzd7*] (10 punkti)
Izmantojot četrus ciparus, iespējams pierakstīt pulksteņa laiku formātā HH:MM. Sastādīt programmu, kas izdrukā ekrānā visus iespējamos pulksteņa laikus, ko var izveidot no lietotāja ievadītajiem četriem cipariem, katru no tiem izmantojot tikai vienu reizi. Nodrošināt, ka lietotājs var ievadīt tikai ciparus.
Piemēram,

- ja lietotājs ievada 1, 2, 3 un 4, programma izdrukā
  12:34 12:43 13:24 13:42 14:23 14:32 21:34 21:43 23:14 23:41
- ja lietotājs ievada 0, 1, 8 un 8, programma izdrukā
  08:01 08:10 18:08

8. uzdevums [*uzd8*] (15 punkti)
Bērni nolēma spēlēt paslēpes. Tā kā neviens nevēlējās būt pirmais, kurš pārējos meklēs, bērni nolēma nostāties aplī un skaitīt skaitāmpantiņu. Katrs bērns (pulksteņrādītāja virzienā) secīgi skaitīja pa vienam skaitāmpantiņa vārdam. Tas, kurš izrunāja skaitāmpantiņa pēdējo vārdu, izstājās no apļa, bet nākamais, kas aplī stāvēja aiz viņa, sāka skaitīt skaitāmpantiņu no jauna. Tā skaitāmpantiņš tika skaitīts, līdz aplī bija palicis tikai viens bērns, tas kuram pirmajam jāmeklē citus bērnus.
Sastādīt programmu, kas nosaka kuram no bērniem pirmajam jāmeklē citus, ja faila *uzd8.in* pirmajā rindā dots skaitāmpantiņš, bet otrajā rindā – ar vienu atstarpi atdalīti bērnu vārdi, ne vairāk kā 50. Bērna vārdu, kuram jāmeklē pirmajam, ierakstīt failā *uzd8.out*.

Piemēram:
ievaddati *uzd8.in*

Ennik, bennik, sikel, sa, sesi viri pagraba. Viens, divi, tris, nu tu esi brivs.
Markuss Laine Marta Juris Santa Gatis Reinis Linda

izvaddati *uzd8.out*

Reinis

9. uzdevums [*uzd9*] (10 punkti)
Biatlona sacensību finālā ar atsevišķo startu tiesības piedalīties bija izcīnījuši *n* sportisti. Pēc tam, kad katrs no dalībniekiem finišēja (sportisti finišēja secībā, kas atbilda viņu kārtas numuriem – skaitlis no 1 līdz *n*), skatītājiem tika paziņots, kādu vietu dotajā brīdī ieņem finišējušais sportists.
Sastādīt programmu, kas nolasa failu *uzd9.in*, kura pirmajā rindā dots skaitlis *n* ($10 \le n \le 30$) – dalībnieku skaits sacensībās, otrajā rindā doti *n* skaitļi, kas atdalīti ar vienu tukšumu, kas norāda sportistu izcīnīto vietu finišēšanas brīdī. Noteikt sacensību rezultātus. Faila *uzd9.out* pirmajā rindā izvadīt pirmā sportista kārtas numuru un pēc vienas atstarpes sacensībās izcīnīto vietu, otrajā rindā otrā sportista kārtas numuru un pēc vienas atstarpes sacensībās izcīnīto vietu, ..., *n*-tajā rindā *n*-tā sportista kārtas numuru un pēc vienas atstarpes sacensībās izcīnīto vietu.

Piemēram,
ievaddati *uzd9.in*

12
1 1 2 3 1 4 4 2 1 3 1 2

izvaddati *uzd9.out*

1 12
2 7
3 8
4 11
5 4
6 10
7 9
8 6
9 3
10 5
11 1
12 2

10. uzdevums [*uzd10*] (10 punkti)

Ralfs skolas kursā "Programmēšanas pamati" šobrīd apgūst tēmu "Saspiešanas algoritmi". Skolotājs viņam ir iedevis programmas piemēru, kas realizē datu saspiešanu pēc RLE algoritma, tas ir, ja sākotnējā simbolu virknē, kāds simbols atkārtojas vairākas reizes, tad saspiestajā simbolu virknē to raksta vienu reizi, un aiz simbola raksta skaitli, kas norāda, cik reizes simbols atkārtojas. Piemēram, simbolu virkni **aaaaabbbacccbdddddd** saspiestā veidā pierakstīta **a5b3ac3bd5**.
Palīdzi Ralfam uzrakstīt programmu, kas no saspiestās simbolu virknes, kas ierakstīta teksta datnē *uzd10.in*, atjauno oriģinālo simbolu virkni un ieraksta to datnē *uzd10.out*.

Skolotāja dotās C++ programmas kods

```
#include <fstream>
using namespace std;
int main(){
  ifstream dati1;
  ofstream dati2;
  dati1.open("uzd10.in",ios::in);
  dati2.open("uzd10.out",ios::out);
  char s1,s2;
  int sk;
  dati1.get(s1);
  while(!dati1.eof()){
     sk=1;
     dati1.get(s2);
     while(s1==s2&&!dati1.eof()){
        sk++;
        dati1.get(s2);
     }
     if(sk==1)
        dati2<<s1;
     else
        dati2<<s1<<sk;
     s1=s2;
  }
  dati1.close();
  dati2.close();
}
```