Liepājas Universitātes

Datorzinātņu olimpiādes 2016

skolēnu programmēšanas konkursa uzdevumi

1. uzdevums [*uzd1*] (5 punkti)

Dots algoritma apraksts strukturētā teksta veidā. Uzzīmē dotā algoritma blokshēmu un sastādi programmu.

Sākums

1. **solis** mainīgajam *a* piešķir sākuma vērtību 2
2. **solis** mainīgajam *b* piešķir sākuma vērtību 3
3. **solis** mainīgajam *c* piešķir izteiksmes *a*+*b* vērtību
4. **solis** izvada mainīgā *a* vērtību
5. **solis** mainīgajam *a* piešķir mainīgā *b* vērtību
6. **solis** mainīgajam *b* piešķir mainīgā *c* vērtību
7. **solis** kamēr mainīgā *c* vērtība ir mazāka par 100, atgriezties pie 3. soļa

Beigas

2. uzdevums [*uzd2*] (5 punkti)

Kārlis vēro robotu, kurš pārvietojas pa taisnu līniju šurpu turpu N soļus. Robota viena soļa garums ir 1 cm. Robots no kustības sākumpunkta jeb nulles punkta pārvietojas K soļus pa labi, pēc tam apgriežas un pārvietojas K soļus atpakaļ līdz nulles punktam, pēc tam atkal apgriežas un pārvietojas K soļus pa labi, un tā visu laiku, līdz kopskaitā veicis N soļus.

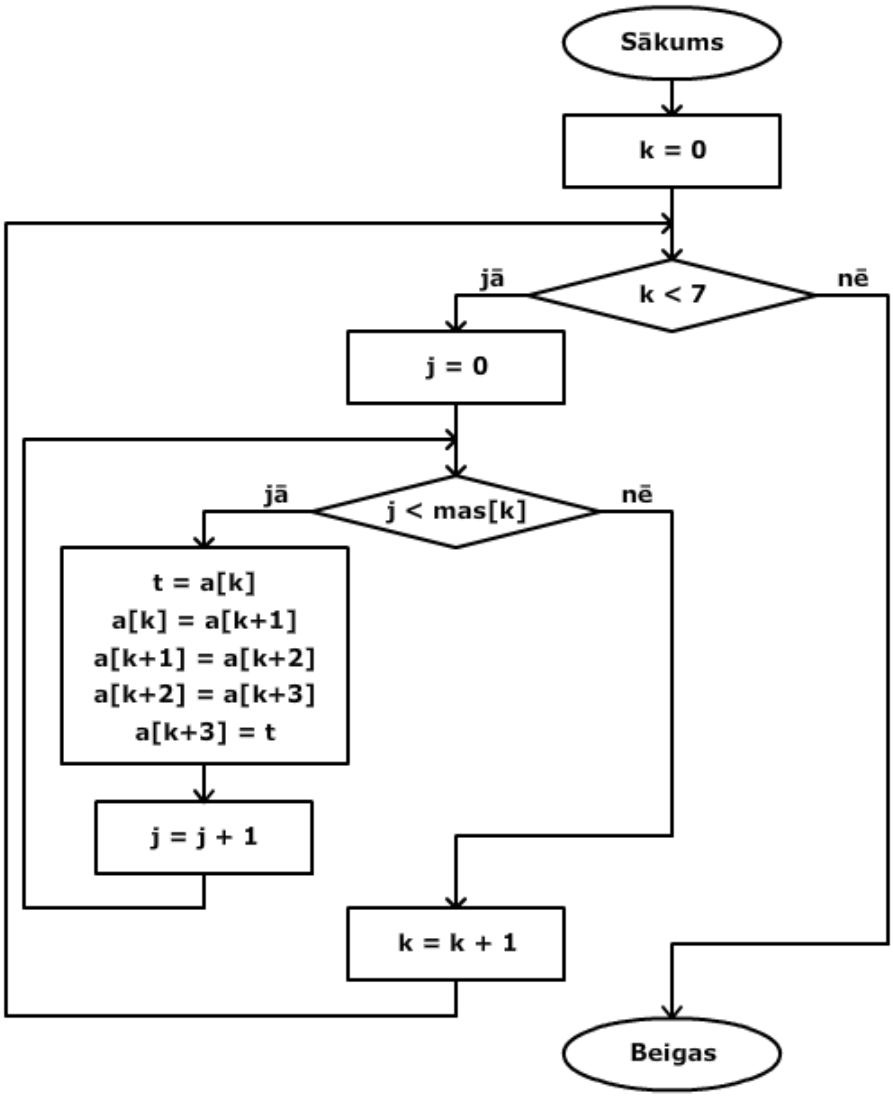
Sastādīt programmu, kas nosaka un izvada ekrānā robota attālumu no nulles punkta, atbilstoši lietotāja ievadītajām N un K vērtībām. N – naturāls skaitlis (0<N<100000), kas apzīmē soļu skaitu, ko veic robots, bet K – naturāls skaitlis (0<K<10000), kas norāda, cik soļus šurpu turpu pārvietojas robots. Nodrošināt, ka lietotājs nevar ievadīt nekorektas N un K vērtības.

3. uzdevums [*uzd3*] (10 punkti)

Sastādi programmu, kas ekrānā izvada dotās tabulas vērtības.

1	9	1	9	1	9	1	9	1
2	8	2	8	2	8	2	8	2
3	7	3	7	3	7	3	7	3
4	6	4	6	4	6	4	6	4
5	5	5	5	5	5	5	5	5
6	4	6	4	6	4	6	4	6
7	3	7	3	7	3	7	3	7
8	2	8	2	8	2	8	2	8
9	1	9	1	9	1	9	1	9

4. uzdevums [*uzd4*] (10 punkti)
Dota algoritma blokshēma. Masīva ***a*** vērtības pirms algoritma izpildes ir {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}, bet pēc izpildes {3, 0, 4, 5, 1, 2, 7, 8, 9, 6}. Pēc blokshēmas noteikt masīva ***mas*** elementu skaitu un vērtības, ja zināms, ka masīva ***mas*** vērtības ir veseli skaitļi intervālā [0;3].



5. uzdevums [*uzd5*] (10 punkti)
Zitai jāmācās reizināšanas tabula, tādēļ viņa uzzīmē rūtiņu tabulu, kuras rindas un kolonnas sanumurē ar skaitļiem no 1 līdz 10, bet tabulas rūtiņās ieraksta atbilstošās rindas un kolonnas numura reizinājumu. Lai vieglāk varētu iegaumēt reizināšanas tabulu, Zita nolēma tabulas rūtiņas izkrāsot. Sākumā viņa visas tabulas rūtiņas nokrāsoja dzeltenas. Otrajā krāsošanas etapā - visas rūtiņas, kas satur pāra skaitļus, pārkrāsoja sarkanas. Trešajā etapā – visas rūtiņas, kas satur skaitļus, kas dalās ar 3, pārkrāsoja zaļas. Pēdējā krāsošanas etapā – visas rūtiņas, kas satur skaitļus, kas dalās ar 5, pārkrāsoja zilas.
Sastādīt programmu, kas saskaita un izvada ekrānā informāciju par to, cik reizināšanas tabulas rūtiņas beigās bija nokrāsotas dzeltenas, cik – sarkanas, cik – zaļas un cik – zilas.

6. uzdevums [*uzd6*] (20 punkti)
Sastādīt programmu, kas ģenerē un izvada ekrānā 5 gadījuma skaitļus intervālā [1;13].

- Ja tiek noģenerēti 5 vienādi skaitļi, izvadīt ekrānā "IMPOSSIBLE", citādi
- ja tiek noģenerēti 4 vienādi skaitļi, izvadīt ekrānā "FOUR OF A KIND", citādi
- ja tiek noģenerēti 3 un 2 vienādi skaitļi, izvadīt ekrānā "FULL HOUSE", citādi
- ja tiek noģenerēti 5 skaitļi, kas veido secīgu skaitļu virkni, izvadīt ekrānā "STRAIGHT", citādi
- ja tiek noģenerēti 3 vienādi skaitļi, izvadīt ekrānā "THREE OF A KIND", citādi
- ja tiek noģenerēti 2 un 2 vienādi skaitļi, izvadīt ekrānā "TWO PAIRS", citādi
- ja tiek noģenerēti 2 vienādi skaitļi, izvadīt ekrānā "ONE PAIR", citādi
- izvadīt ekrānā "NOTHING".

7. uzdevums [*uzd7*] (5 punkti)

Doti taisnstūra veida aploksnes un apsveikuma kartiņas izmēri. Sastādīt programmu, kas nolasa aploksnes un apsveikuma kartiņas izmērus no faila *uzd7.in* un nosaka, vai apsveikuma kartiņu, nelokot un neapgriežot tās malas, var ievietot aploksnē. Ja var, tad failā uzd7.out izvadīt "VAR", ja nevar, tad – "NEVAR". Faila *uzd7.in* pirmajā rindā doti divi naturāli skaitļi N un M – aploksnes izmēri, kas atdalīti ar atstarpi, bet otrajā rindā doti divi naturāli skaitļi K un L – apsveikuma kartiņas izmēri, kas atdalīti ar atstarpi. Neviens no dotajiem skaitļiem nav mazāks par 1 un lielāks par 100.

Piemēram,

ievaddati *uzd7.in*	izvaddati *uzd7.out*
4 6 5 4	VAR

8. uzdevums [*uzd8*] (10 punkti)

Agrim ar draugiem patīk apmeklēt kino. Visiem zināms, ka kino apmeklēt interesantāk ir lielā draugu kompānijā. Tādēļ Agris, iegādājoties kino biļetes, kasierim vienmēr pajautā, kāds ir lielākais vienā rindā blakus esošu brīvo vietu skaits?

Sastādīt programmu, kas nolasa faila *uzd8.in* datus un aprēķina, un izvada failā *uzd8.out* lielāko blakus esošu brīvo vietu skaitu vienā rindā. Faila *uzd8.in* pirmajā rindā doti divi naturāli skaitļi N un M, kas atdalīti ar atstarpi; N – rindu skaits kinoteātra zālē, M – vietu skaits vienā rindā, ($1 \le n, m \le 100$). Katrā no nākamajām N rindām doti M cipari – 0 vai 1, kas savstarpēji atdalīti ar atstarpi; 1 norāda aizņemtu vietu, 0 – brīvu vietu.

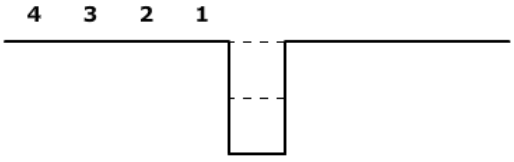
Piemēram,

ievaddati *uzd8.in*	izvaddati *uzd8.out*
4 6 1 1 1 0 0 0 0 1 1 0 1 1 1 0 0 0 0 1 1 0 0 1 0 1	4

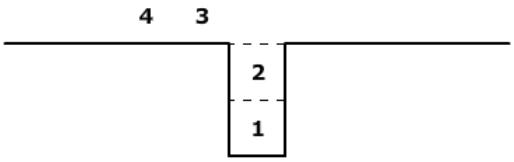
9. uzdevums [*uzd9*] (15 punkti)

Eži cits aiz cita iet pa meža taku. Ik pa laikam tiem ceļā gadās truša alas, kurās tie iekrīt. Alas ir tik dziļas, ka tajās dažkārt iekrīt pat vairāki eži vienlaicīgi – viens virs otra. Kad truša ala ir aizpildījusies ar iekritušajiem ežiem, pārējie eži pāriet tai pāri. Pēc tam gājienu noslēdzošais ezis izvelk no alas vienu ezi, izvilktais ezis izvelk no alas nākamo, tas – nākamo, un tā turpina, līdz alā ežu vairāk nav. Pēc tam ežu gājiens turpinās; ežu secība ir mainījusies. Piemēram, ja pa taku iet 4 eži un tiem ceļā gadās truša ala, kurā var iekrist divi eži, tad alas šķērsošanu var attēlot šādi:

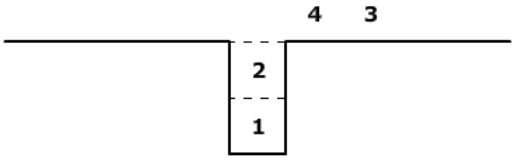
1) Eži pienāk pie alas



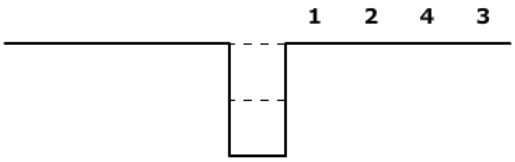
2) Divi eži iekrīt alā



3) Divi eži šķērso alu



4) Eži izvelk savus biedrus no alas un turpina ceļu



Sastādīt programmu, kas nolasa failu *uzd9.in*, kura pirmajā rindā doti divi skaitļi N un M, kas atdalīti ar vienu atstarpi, kur N - ežu skaits un M – truša alu skaits. Faila otrajā rindā doti M skaitļi, kas atdalīti ar vienu atstarpi – K_1, K_2, K_3, ..., K_M – truša alu dziļumi, kuras secīgi šķērso eži. Failā *uzd9.out* izdrukāt N skaitļus vienā rindā, kas atdalīti ar vienu atstarpi un kas norāda ežu secību pēc truša alu šķērsošanas.

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

Harijam skolas kursā "Programmēšanas pamati" jāapgūst tēma "Datu šifrēšana". Skolotājs viņam ir iedevis programmas piemēru, kas realizē datu šifrēšanu, izmantojot Džona Falkonera (John Falconer) 1685. gadā publicēto "Divu burtu alfabētu".

Piemēram,

ja šifrējamais teksts ir: **KLASES VAKARS**, tad šifrētā simbolu virknes ir:

ABAABABABAAAAAABAAABAABAABAAAB BAABBAAAAAABAABAAAAABAAAABAAAB.

Palīdzi Harijam uzrakstīt programmu, kas nolasa ar "Divu burtu alfabētu" šifrētu tekstu no faila *uzd10.in*, atšifrē to, un rezultātu ieraksta failā *uzd10.out*.

"Divu burtu alfabēta" algoritma realizācija (Java kods)

```
import java.io.*;
public class DivuBurtuAlfabets {
  public static void main(String[] args) {
    try{
      char s;
      int x;
      BufferedReader dati1=new BufferedReader
                    (new FileReader("uzd10.in"));
      BufferedWriter dati2=new BufferedWriter
                    (new FileWriter("uzd10.out"));
      x=dati1.read();
      s=(char)x;
      while(x!=-1){
        switch(s){
          case 'A': dati2.write("AAAAA"); break;
          case 'B': dati2.write("AAAAB"); break;
          case 'C': dati2.write("AAABA"); break;
          case 'D': dati2.write("AAABB"); break;
          case 'E': dati2.write("AABAA"); break;
          case 'F': dati2.write("AABAB"); break;
          case 'G': dati2.write("AABBA"); break;
          case 'H': dati2.write("AABBB"); break;
          case 'I': dati2.write("ABAAA"); break;
          case 'K': dati2.write("ABAAB"); break;
          case 'L': dati2.write("ABABA"); break;
          case 'M': dati2.write("ABABB"); break;
          case 'N': dati2.write("ABBAA"); break;
          case 'O': dati2.write("ABBAB"); break;
          case 'P': dati2.write("ABBBA"); break;
          case 'R': dati2.write("BAAAA"); break;
          case 'S': dati2.write("BAAAB"); break;
          case 'T': dati2.write("BAABA"); break;
          case 'V': dati2.write("BAABB"); break;
          case 'Z': dati2.write("BABBB"); break;
          default: dati2.write(s);
        }
        x=dati1.read();
        s=(char)x;
      }
      dati1.close();
      dati2.close();
    } catch (IOException e){}
  }
}
```

"Divu burtu alfabēta" algoritma realizācija (C++ 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 s;
  dati1.get(s);
  while(!dati1.eof()){
    switch(s){
      case 'A': dati2<<"AAAAA"; break;
      case 'B': dati2<<"AAAAB"; break;
      case 'C': dati2<<"AAABA"; break;
      case 'D': dati2<<"AAABB"; break;
      case 'E': dati2<<"AABAA"; break;
      case 'F': dati2<<"AABAB"; break;
      case 'G': dati2<<"AABBA"; break;
      case 'H': dati2<<"AABBB"; break;
      case 'I': dati2<<"ABAAA"; break;
      case 'K': dati2<<"ABAAB"; break;
      case 'L': dati2<<"ABABA"; break;
      case 'M': dati2<<"ABABB"; break;
      case 'N': dati2<<"ABBAA"; break;
      case 'O': dati2<<"ABBAB"; break;
      case 'P': dati2<<"ABBBA"; break;
      case 'R': dati2<<"BAAAA"; break;
      case 'S': dati2<<"BAAAB"; break;
      case 'T': dati2<<"BAABA"; break;
      case 'V': dati2<<"BAABB"; break;
      case 'Z': dati2<<"BABBB"; break;
      default: dati2<<s;
    }
    dati1.get(s);
  }
  dati1.close();
  dati2.close();
}
```