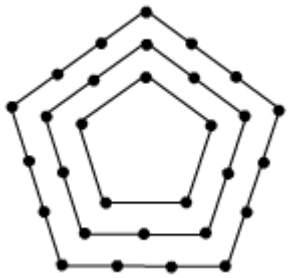
Liepājas Universitātes
**Datorzinātņu olimpiādes 2018**
programmēšanas konkursa uzdevumi studentiem

**1. uzdevums [uzd1]** (5 punkti)
Ir 2500.gada sākums. Uz dzīvi Robotaunā pārcēlušies N jauni roboti, kuri turpmāk dzīvos pēc sekojošiem likumiem:

- katra gada sākumā visi roboti tiek sadalīti grupās pa 4 robotiem;
- viena gada laikā 4 robotu grupa uzbūvē 5 jaunus robotus;
- katra gada beigās robotus, kuri sasnieguši 3 gadu vecumu, izslēdz un nodod pārstrādei.

Uzzīmēt blokshēmu, pēc kuras var noteikt, cik roboti dzīvos Robotaunā pēc K gadiem.

**2. uzdevums [uzd2]** (5 punkti)
Tornim ir K stāvi. Katram tā stāvam ir regulāra N-stūra forma. Torņa augšējam stāvam katrā sienā ir divas šaujamlūkas (katrā stūrī pa vienai), bet katra nākamā stāva sienā ir par 1 šaujamlūku vairāk nekā iepriekšējā stāva sienā. Sastādi programmu, kas nosaka un izvada ekrānā, cik tornī šaujamlūku, ja lietotājs ievada K un N vērtību. Nodrošināt, ka lietotājs K vērtību var ievadīt tikai intervālā [1;1000], bet N vērtību intervālā [3;1000].



3 stāvu 5-stūra formas torņa shematisks zīmējums skatā no augšas

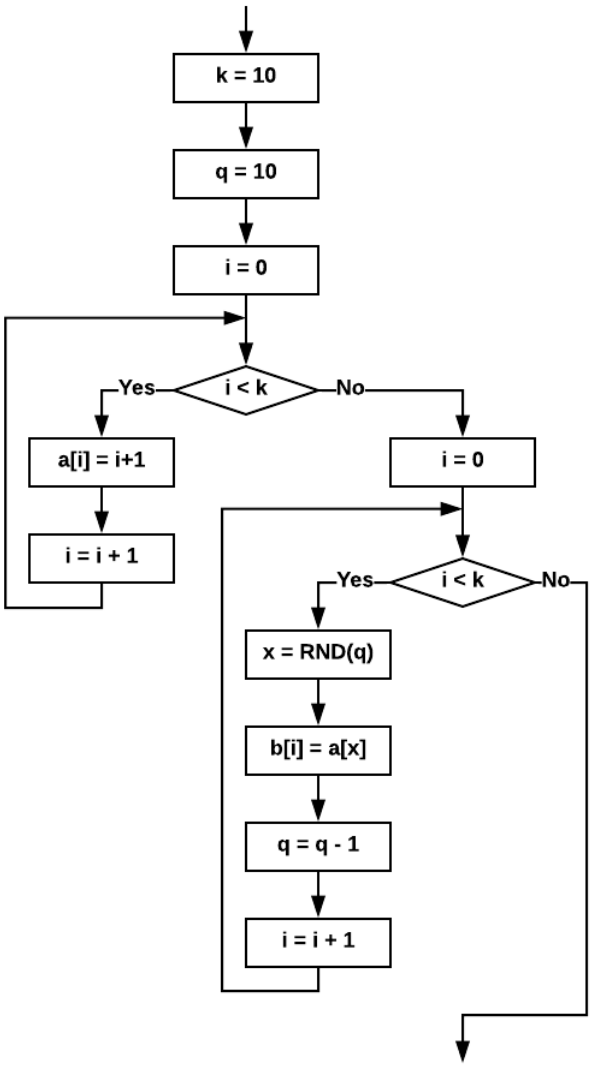
**3. uzdevums [uzd3]** (10 punkti)
Atrodi sakarības starp skaitļiem dotās n x m tabulas rūtiņās. Sastādi programmu, kas, izmantojot atrastās sakarības, izvada dotos skaitļus ekrānā tabulas formā (bez atdalošajām līnijām). Lielumu n un m vērtības ievada lietotājs. Lieluma n vērtība var būt tikai vesels skaitlis intervālā [3;21], kas dalās ar 3, bet m vērtība var būt tikai vesels pāra skaitlis intervālā [2;20]. Piemēram, ja n = 9 un m=8, tad tabula tiek aizpildīta ar šādām vērtībām:

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

**4. uzdevums [uzd4]** (10 punkti)
Sastādīt programmu, kas izvada ekrānā korekti pierakstītu algebrisku izteiksmi a+bx+cy jebkurām lietotāja ievadītām skaitliskām a, b un c vērtībām. Piemēram, ja lietotājs ievada a=2, b=0 un c=3, tad programma izvada ekrānā izteiksmi 2+3y; ja lietotājs ievada a=3, b=-1 un c=0, tad programma izvada 3-x.

**5. uzdevums [uzd5]** (10 punkti)
Dots algoritma blokshēmas fragments. Ar RND(q) blokshēmā apzīmēta funkcija, kuras vērtība ir nejaušs vesels skaitlis intervālā [0;q). Pēc blokshēmas nosaki:

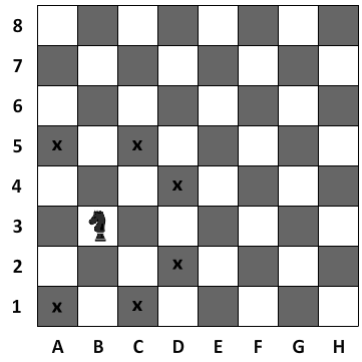
- cik vērtības ir masīvā a;
- cik vērtības ir masīvā b;
- kādas ir masīva a vērības;
- kādas ir masīva b vērtības;
- kādu problēmu/uzdevumu risina dotais blokshēmas fragments?




**6. uzdevums [uzd6]** (10 punkti)
Viesnīcā ierodas kaprīzs klients un vēlas apmesties istabā, kura atrodas vistuvāk liftam. Ja brīvas ir vairākas istabas, kuras atrodas vienādā attālumā no lifta, tad – tajā, kura atrodas zemākā stāvā. Zināms, ka viesnīcas istabas kārtas numurs sastāv no diviem cipariem, pirmais cipars apzīmē stāvu, kurā istaba atrodas, otrais norāda istabas attālumu no lifta. Viesnīcai ir 9 stāvi un katrā no tiem ir 9 istabas.
Sastādīt programmu, kas no faila *uzd6.in* nolasa brīvo viesnīcas istabu numurus (istabu numuri doti vienā rindā, tie atdalīti ar vienu atstarpi) un sarindo tos klientam vēlamajā secībā, sākumā viesnīcas istabas numurs, kurš visprecīzāk atbilst klienta vēlmēm, beigās istabas numurs, kurš vismazāk atbilst klienta vēlmēm. Sakārtotos viesnīcas istabu numurus izdrukāt failā *uzd6.out* (vienā rindā atdalītus ar atstarpi).
Piemēram,

| ievaddati *uzd6.in* | izvaddati *uzd6.out* |
|---|---|
| 12 25 11 43 22 15 18 31 44 52 | 11 31 12 22 52 43 44 15 25 18 |

**7. uzdevums [uzd7]** (15 punkti)




Zināms, ka šaha dēlītis sastāv no 8x8 melnbaltām rūtiņām. Katrai rūtiņai ir savas koordinātes, kuras sastāv no lielā burta un cipara. Katrai šaha figūrai ir stingri noteikts, kā to viena gājiena laikā drīkst pārvietot. Piemēram, zirgs viena gājiena laikā var tikt pārvietots jebkurā virzienā divus soļus uz priekšu un vienu pa labi vai kreisi, katrā solī pārejot uz blakus esošo lauciņu (diviem blakus esošiem lauciņiem ir kopīga mala, t.i., zirgs nedrīkst pārvietoties diagonāļu virzienos).

Sastādīt programmu, kas, zinot zirga atrašanās vietas koordinātes šaha laukumā, nosaka, uz kādas krāsas lauciņa zirgs atrodas un uz kādu koordinātu lauciņiem zirga figūriņa varētu tikt pārvietota viena spēles gājiena laikā.

Faila uzd7.in vienīgajā rindā dotas zirga atrašanās vietas koordinātas, piemēram, B3 (skat. attēlu).

Failā uzd7.out pirmajā rindā jāizvada BALTS, ja zirga figūra atrodas uz baltā spēles lauciņa vai MELNS, ja uz melnā. Katrā nākošajā faila uzd7.out rindā brīvi izraudzītā secībā jāizvada visas iespējamās zirga koordinātes pēc viena gājiena, piemēram,

BALTS
A5
C5
D4
D2
C1
A1

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

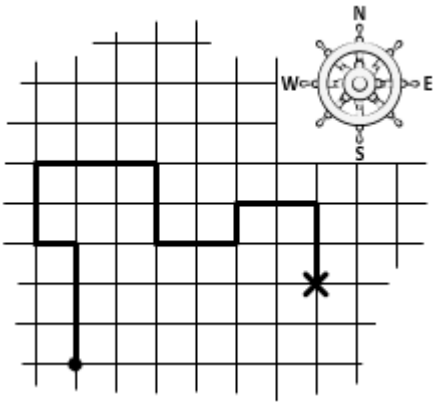
Dārgumu meklētāji, vadoties pēc vecas pirātu kartes, ir nokļuvuši neapdzīvotā salā un uzslējuši pagaidu apmetni, lai pārlaistu nakti. Agri no rīta, vadoties pēc pirātu norādījumiem, dārgumu meklētāji dodas meklēt apslēptos dārgumus. Tikai vēlā pēcpusdienā pārguruši, laužoties cauri aizaugušiem džungļiem un šķērsojot purvājus, viņi nokļūst līdz dārgumiem. Tā kā dārgumu lāde ir ļoti smaga, viņi nevēlas mērot atpakaļceļu līdz apmetnei pa pirātu norādīto ceļu, bet pa taisno.

Sastādi programmu, kas no faila uzd8.in nolasa pirātu atstāto instrukciju, kur katrā rindā dots viens lielais burts un viens vesels skaitlis robežās no 1 līdz 1000, kas atdalīti ar atstarpi, burts apzīmē kustības virziens (N – uz ziemeļiem, E – uz austrumiem, S – uz dienvidiem, W – uz rietumiem), bet skaitlis – soļu skaitu, kurš jāveic dotajā virzienā, un aprēķina un izvada failā aptuvenu (precizitāte ir atkarīga no izvēlētās programmēšanas valodas) īsāko attālumu soļos no dārgumu atrašanās vietas līdz apmetnei.

Piemēram, ja failā uzd8.in dota instrukcija

N3
W1
N2
E3
S2
E2
N1
E2
S2

tad programmai failā uzd8.out jāizvada aptuveni 6.324555



**9. uzdevums [uzd9]** (15 punkti)
Faila uzd9.in vienīgajā rindā dots vesels skaitlis, par kuru ir zināms, ka tam vienmēr ir pāra skaits ciparu un ka tas sastāv vismaz no 2 cipariem, bet ne vairāk kā no 100000 cipariem. Sastādi programmu, kas pārkārto dotā skaitļa ciparus pēc sekojošiem nosacījumiem:

- sākumā skaitļa ciparus sadala divās vienādās daļās,
- tad katras daļas ciparus apmaina vietām – apgriež otrādi,
- tad atkārto katras daļas ciparu apgriešanu, neņemot vērā skaitļa pirmo un pēdējo ciparu,
- pēc tam, atkārto katras daļas ciparu apgriešanu, neņemot vērā skaitļa pirmos divus un pēdējos divus ciparus,
- līdzīgā veidā apgriešanas darbību atkāto līdz katrā skaitļa daļā apgriežamo ciparu skaits ir samazinājies līdz 1.

Jauniegūto skaitli izvadīt failā uzd9.out.
Piemēram, ja failā uzd9.in dots skaitlis 1234554321, tad programmai jāveic sekojoša skaitļu ciparu pārkārtošana

12345|54321
54321|12345
51234|43215
51432|23415
51423|32415

un failā uzd9.out jāizvada 5142332415.

**10. uzdevums [uzd10]** (10 punkti)
Kurts skolā apgūst mācību priekšmetu "Programmēšanas pamati". Klasē, strādājot kopā ar skolotāju, Kurts ir uzrakstījis programmu, kas aprēķina divu lielu veselu skaitļu summu.
Pārveido Kurta uzrakstīto programmu, tā, lai tā nolasa no faila uzd10.in divus lielus veselus skaitļus, katrs skaitlis dots savā rindā, un aprēķina šo skaitļu starpību. Aprēķināto skaitļu starpību, jāizvada failā uzd10.out.

**C++ kods divu skaitļu summas aprēķinam**

```
#include <fstream>
using namespace std;
int main(){
	ifstream dati1;
	ofstream dati2;
	dati1.open("uzd10.in",ios::in);
	dati2.open("uzd10.out",ios::out);
	string sk1, sk2;
	int n1, n2, n3;
	int i;
	dati1>>sk1;
	dati1>>sk2;
	n1=sk1.length();
	n2=sk2.length();
	if(n1>n2) n3=n1; else n3=n2;
	int x1[n3+1]={0};
	int x2[n3+1]={0};
	int x3[n3+1]={0};
	for(i=0; i<n1; i++) x1[i]=sk1[n1-i-1]-'0';
	for(i=0; i<n2; i++) x2[i]=sk2[n2-i-1]-'0';
	for(i=0; i<n3; i++){
		x3[i+1]=(x1[i]+x2[i]+x3[i])/10;
		x3[i]=(x1[i]+x2[i]+x3[i])%10;
	}
	if(x3[n3]!=0) dati2<<x3[i];
	for(i=n3-1; i>=0; i--) dati2<<x3[i];
	dati1.close();
	dati2.close();
}
```

**Java kods divu skaitļu summas aprēķinam**

```
import java.io.*;
public class Sum {
      public static void main(String[] args) {
      try{
            BufferedReader dati1=new BufferedReader(new FileReader("uzd10.in"));
            BufferedWriter dati2=new BufferedWriter(new FileWriter("uzd10.out"));
            String sk1, sk2;
            int n1, n2, n3;
            int i;
            sk1=dati1.readLine();
            sk2=dati1.readLine();
            n1=sk1.length();
            n2=sk2.length();
            n3=Math.max(n1,n2);
            int[] x1=new int[n3+1];
            int[] x2=new int[n3+1];
            int[] x3=new int[n3+1];
            for(i=0; i<n1; i++) x1[i]=sk1.charAt(n1-i-1)-'0';
            for(i=0; i<n2; i++) x2[i]=sk2.charAt(n2-i-1)-'0';
            for(i=0; i<n3; i++){
                  x3[i+1]=(x1[i]+x2[i]+x3[i])/10;
                  x3[i]=(x1[i]+x2[i]+x3[i])%10;
            }
            if(x3[n3]!=0) dati2.write(x3[i]+"");
            for(i=n3-1; i>=0; i--) dati2.write(x3[i]+"");
            dati1.close();
            dati2.close();
      } catch (IOException e){System.out.print("Nevar atrast failu!");}
      }
}
```

**Python3 kods divu skaitļu summas aprēķinam**

```
dati1 = open("uzd10.in", "r")
dati2 = open("uzd10.out", "w")
n1=0; n2=0; n3=0
i=0
sk1 = dati1.readline()
sk2 = dati1.readline()
n1=len(sk1)-1
n2=len(sk2)
if n1>n2: n3=n1
else: n3=n2
x1 = [0] * (n3+1)
x2 = [0] * (n3+1)
x3 = [0] * (n3+1)
for i in range(n1): x1[i]=int(sk1[n1-i-1])
for i in range(n2): x2[i]=int(sk2[n2-i-1])
for i in range(n3):
      x3[i+1]=int((x1[i]+x2[i]+x3[i])/10)
      x3[i]=(x1[i]+x2[i]+x3[i])%10
if x3[n3] != 0: dati2.write(str(int(x3[i])))
for i in range(n3-1, -1, -1): dati2.write(str(int(x3[i])))
dati1.close()
dati2.close()
```

**Php kods divu skaitļu summas aprēķinam**

```
<?php
	$dati1=fopen('uzd10.in','r');
	$dati2=fopen('uzd10.out','w');
	$sk1=fgets($dati1);
	$sk2=fgets($dati1);
	$sk1=trim($sk1);
	$sk2=trim($sk2);
	$sk1=strrev($sk1);
	$sk2=strrev($sk2);
	$n1=strlen($sk1);
	$n2=strlen($sk2);
	if($n1>$n2) $n3=$n1; else $n3=$n2;
	$x1=array_fill(0,$n3+1,"0");
	$x2=array_fill(0,$n3+1,"0");
	$x3=array_fill(0,$n3+1,"0");
	for($i=0; $i<$n1; $i++) $x1[$i]=$sk1[$i];
	for($i=0; $i<$n2; $i++) $x2[$i]=$sk2[$i];
	for($i=0; $i<$n3; $i++) {
		$x3[$i+1]=(int)((intval($x1[$i])+intval($x2[$i])+intval($x3[$i]))/10);
		$x3[$i]=(intval($x1[$i])+intval($x2[$i])+intval($x3[$i]))%10;
	}
	if(intval($x3[$n3])!=0) fwrite($dati2,$x3[$n3]);
	for($i=$n3-1; $i>=0; $i--) fwrite($dati2,$x3[$i]);
	fclose($dati1);
	fclose($dati2);
?>
```