Liepājas Universitātes
Datorzinātņu olimpiāde 2019
programmēšanas konkursa uzdevumi studentiem

Task 1 [uzd1] (5 points)
Write a program that determines (a) whether a brick with dimensions **AxBxC** can be put through a rectangular hole with dimensions **NxM** and (b) whether a cylinder with a height **H** and a basic radius **R** can be put through a rectangular hole with dimensions **NxM**. All dimensions are given in centimeters. The values of all variables, input by user, are integers between 1 and 1000. The program must ensure that the user cannot enter incorrect input values. The output of the program must be two sentences – (a) "CAN/CANNOT insert a brick into the hole", and (b) "CAN/CANNOT insert the cylinder into the hole".

Task 2 [uzd2] (5 points; or 7 points if the code of your program has not loops)
Write a program that calculates and output the value of the N-th element of the given integer sequence
1, 2, 3, 6, 5, 10, 7, 14 …

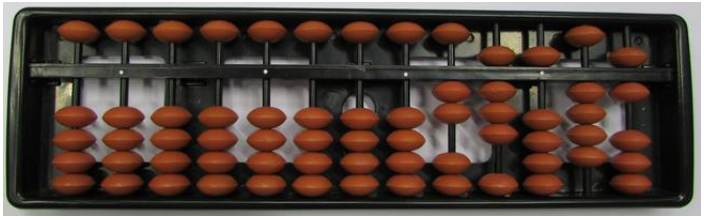
The value of the variable N must be input by user and it must be integer between 1 and 10000. The program must ensure that the user cannot enter incorrect input values.

Task 3 [uzd3] (10 points)
Find relationships between the numbers in the given **N x N** table. Write the program that, applying the found relationships, output the given numbers on the screen in tabular form (without separating lines). The value of the variable N must be input by user and it can only be an integer in the interval [3; 20].

For example: if N = 9 than you will get the following table:

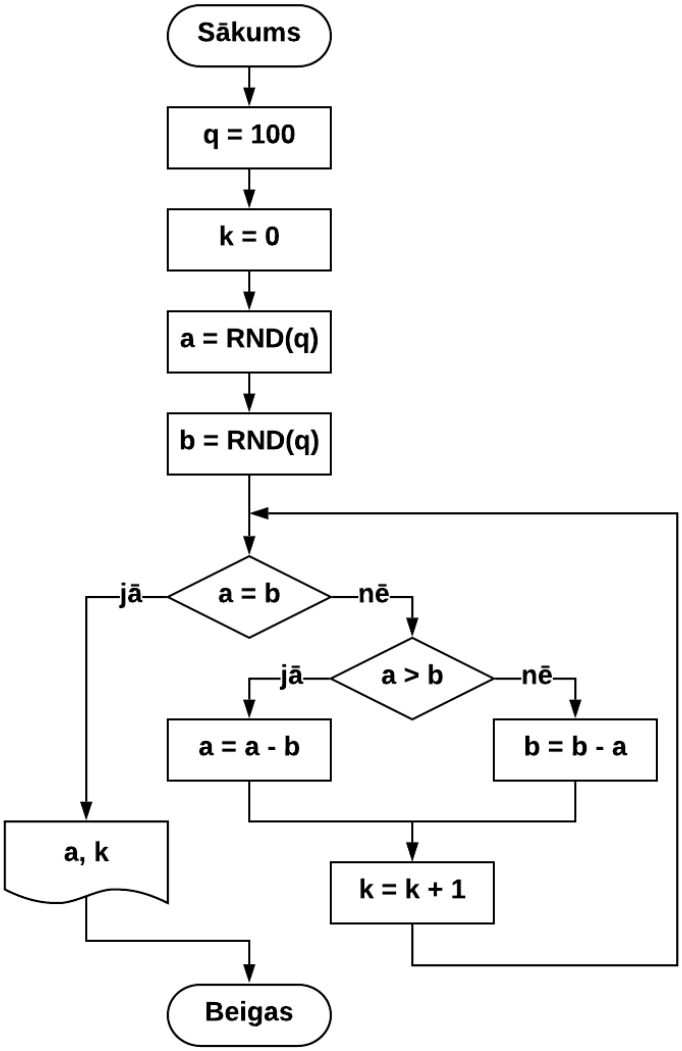
1	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0
1	2	1	0	0	0	0	0	0
1	3	3	1	0	0	0	0	0
1	4	6	4	1	0	0	0	0
1	5	10	10	5	1	0	0	0
1	6	15	20	15	6	1	0	0
1	7	21	35	35	21	7	1	0
1	8	28	56	70	56	28	8	1

Task [uzd4] (10 points)
The figure represents soroban, i.e., Japanese abacus, displaying value 28546. Write the program that outputs representation of integer N, input by user, as on the soroban (see example in figure). The length of the N does not exceed 30 digits.


N=28546

```
0 | | 0 |
| 0 0 | 0
---------
0 0 | 0 0
0 0 0 0 |
| 0 0 0 0
0 | 0 0 0
0 0 0 | 0
```

Task 5 [uzd5] (5 points)
Write program that implements the algorithm defined by the given flowchart. RND(q) means function that returns random integer in interval [1;q].



Task 6 [uzd6] (10 points)
Alex has found an open notebook on the brother's desk. He has noticed there a description of the algorithm for processing of digit sequence. He decided to apply it to a digital string 112233445566778899 and got 36 as a result.

Start
Step 1 – Selects a digit string
Step 2 – Remove all digits located in positions with even index
Step 3 – Write the sum of the deleted digits at the end of the string of remaining digits
Step 4 – Remove all digits located in positions with odd index
Step 5 – Write the sum of the deleted digits at the end of the string of remaining digits.
Step 6 – If more than two digits has remained in the digit string, the go to Step2,
else go to Step 7.
Step 7 – Result: the last two digits
Finish

Write program that applies the given algorithm for the digit string read from text file u*zd6.in*; the length of the string does not exceed 250 digits. The program must output the result into text file *uzd6.out*. Note – if the first digit of the final string is 0, it must be omitted.

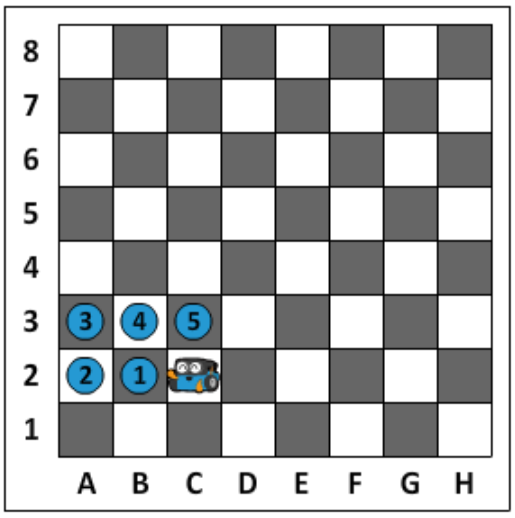
For example,

input *uzd6.in*	output *uzd6.out*
11020234567	49

Task 7 [uzd7] (15 points)

The robot moves along the squares of the chess field. It leaves one ball on each square after crossing it. Each ball has unique identification (ID), i.e., integer. The balls have been put on squares in ascending order of their ID. The smallest ID is 1. The robot moves according to the following rules:

- At the beginning, the robot is placed on a blank chess board in a randomly selected square facing the letter that denotes the column of the chess board;
- If there is no ball to the right of the robot, the robot turns right and moves one step (square) forward;
- If there is a ball to the right of the robot, the robot moves one step (square) forward;
- The robot stops moving either if it cannot make a move anymore or if all chess squares have balls.



Write program that read coordinates of the starting square from text file *uzd7.in*. The result of the program must be output in two lines in text file *uzd7.out* – (a) the first line must contain two integers – coordinates of the starting square, (2) the second line must contain one integer – index of the last settled ball.

For example,

input *uzd7.in*	output *uzd7.out*
B2	A1 8

Task 8 [uzd8] (10 points, if the code of the program does not contain array;
-5 points, if the code of the program contains an array)

In some sports, the performance of athletes is judged by several judges. Each individual judge awards some score value. The highest and lowest values are then discarded, and finally the average of the remaining values is calculated. If the highest or lowest value have awarded by several judges, the all highest or lowest values are discarded.

Write program that reads scores awarded by the judges from plain text file *uzd8.in*. The file can contain no more than 100 scores. Each score is positive integer recorded in separate line. As the result, the program calculates average score and outputs it in plain text file *uzd8.out*. (The accuracy of the average score depends on the selected programming language.)

For example,

input *uzd8.in*	output *uzd8.out*
5 7 3 2 4 7 6	5.4

Task 9 [uzd9] (15 points, if the code of the program does not contain two dimensional array;
-5 points, if the code of the program contains two-dimensional array)

Alex is once again interested in records of his brother's notebook. There he has found the following:

Vigenère cipher

A method of encrypting alphabetic text, based on the letters of a keyword. It is a form of polyalphabetic substitution. First described in 1553. To encrypt, a table of alphabets is used, termed Vigenère square or Vigenère table. It has the alphabet written out 26 times in different rows, each alphabet shifted one position to the left compared to the previous alphabet.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	**O**	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	**G**	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Each row starts with a key letter. The rest of the row holds the letters A to Z (in shifted order). For successive letters of the message, successive letters of the key string will be taken and each message letter enciphered by using its corresponding key row. The next letter of the key is chosen, and that row is gone along to find the column heading that matches the message character. The letter at the intersection of [key-row, msg-col] is the enciphered letter.

Suppose that the plain text to be encrypted is SKOLAS SOMA and the keyword used is OLA. Then, the resulting cryptogram is GVOZLS GZMO.

Note that the same letter in input text can be can be replaced by different letters, in sample case – A is replaced by L and O. And, two different letters can be replace by identical letter, in sample case – O and A are replaced by O.

Write program that read input data from plain text file *uzd9.in*. The first line of the text contains keyword, and the second line contains text to be encrypted. All text is capitalized. The program must encrypt the input text using Vigenère cipher and without using a two-dimensional array. The encrypted text must be written in plain text file *uzd9.out*.

For example,

input *uzd9.in*
OLA
SKOLAS SOMA

output *uzd9.out*
GVOZLS GZMO

Task 10 [uzd9] (10 points)

Kurts has been learning the basics of programming at school. During the class, Kurts has written a program that calculates the sum of two large integers. Modify the program written by Kurts. It must read two large integers from the text file *uzd10.in*, each integer has been written in its own line. The program must calculate the difference between the input numbers and write the result in the text file *uzd10.out*.

C++ code for the calculation of the sum of two large integers

```
#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();
}
```

Python3 code for the calculation of the sum of two large integers

```
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()
```

Java code for the calculation of the sum of two large integers

```
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!");}
      }
}
```

PHP code for the calculation of the sum of two large integers

```
<?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);
?>
```