Олимпиада «Ломоносов» по информатике. 2022-2023 учебный год
Отборочный этап 7-9 класс
Задачи, решения, критерии

Задача 1. Супердробь

Первый вариант условия: На стандартном потоке ввода вводятся 8 целых чисел A, B, C, D. Каждое из чисел не меньше -2147483648 и не больше 2147483647. Найдите такую перестановку K, L, M, N этих чисел, что выражение $K/L + M/N$ будет максимально возможным среди всех перестановок заданных чисел. В качестве ответа на стандартный поток вывода выведите два целых числа P и Q, таких что $Q > 0$, P/Q является несократимой рациональной дробью и $P/Q = K/L + M/N$.

Пример:

```
Ввод:
1 2 3 4
Вывод:
11 2
```

Второй вариант условия: На стандартном потоке ввода вводятся 8 целых чисел A, B, C, D. Каждое из чисел не меньше -2147483648 и не больше 2147483647. Найдите такую перестановку K, L, M, N этих чисел, что выражение $K/L + M/N$ будет минимально возможным среди всех перестановок заданных чисел. В качестве ответа на стандартный поток вывода выведите два целых числа P и Q, таких что $Q > 0$, P/Q является несократимой рациональной дробью и $P/Q = K/L + M/N$.

Пример:

```
Ввод:
1 2 3 4
Вывод:
5 6
```

Решение задачи 1

Код возможного решения для первого варианта условия

```
#! /bin/python3

import sys
import itertools
import math

numbers = [int(x) for x in sys.stdin.read().split()]
permutations = list(itertools.permutations(numbers))

has_max = False
max_p = 0
max_q = 0
for perm in permutations:
    p = perm[0] * perm[3] + perm[2] * perm[1]
    q = perm[1] * perm[3]
    if q == 0:
        continue
    d = math.gcd(p, q)
    p //= d
    q //= d
    if q < 0:
        p = -p
        q = -q
    if not has_max:
        has_max = True
        max_p, max_q = p, q
    else:
```

```
        # p / q - max_p / max_q
        pp = p * max_q - max_p * q
        qq = q * max_q
        if pp > 0:
            has_max = True
            max_p, max_q = p, q

if has_max:
    print(max_p, max_q)
```

Код возможного решения для второго варианта условия

```
#! /bin/python3

import sys
import itertools
import math

numbers = [int(x) for x in sys.stdin.read().split()]
permutations = list(itertools.permutations(numbers))

has_min = False
min_p = 0
min_q = 0
for perm in permutations:
    p = perm[0] * perm[3] + perm[2] * perm[1]
    q = perm[1] * perm[3]
    if q == 0:
        continue
    d = math.gcd(p, q)
    p //= d
    q //= d
    if q < 0:
        p = -p
        q = -q
    if not has_min:
        has_min = True
        min_p, min_q = p, q
    else:
        # p / q - min_p / min_q
        pp = p * min_q - min_p * q
        qq = q * min_q
        if pp < 0:
            has_min = True
            min_p, min_q = p, q

if has_min:
    print(min_p, min_q)
```

Критерий оценивания решений задачи 1

Общим критерием, который не зависит от варианта формулировки условий, является количество тестов, верно пройденных программой, составленной участником. Высчитывается доля (в процентах) таких тестов по отношению к общему количеству тестов. Полученное количество процентов, округлённое до целого, определяет количество технических баллов за решение.

Задача 2. Прайморадичная система счисления

В задаче рассмотрим прайморадичную систему счисления, в которой будем записывать неотрицательные целые числа. В привычной позиционной системе, например, десятичной запись числа основывается на единственном его представлении в виде суммы попарных произведений цифр на степени 10: $d_n d_{n-1} \dots d_i \dots d_2 d_1 = d_n * 10^{n-1} + d_{n-1} * 10^{n-2} + \dots + d_i * 10^{i-1} + \dots + d_2 * 10^1 + d_1 * 10^0$, $0 \leq d_k \leq 10 - 1$. В смешанных системах счисления похожее представление имеет место, но используются не степени, а другие последовательности – числа Фибоначчи, факториалы и прочее.

Прайморадичная система счисления – смешанная, и в ней основаниями являются праймориалы $p_i\#$: 1, 2, 6, 30, 210, 2310, … , где $i = 1, 2, 3, \dots$. Каждый праймориал является произведением вроде факториала, но множителями в нём являются подряд идущие простые числа: $p_n\# = 1 * 2 * \dots * p_i * \dots * p_n$, где p_i – i-ое простое число, а $p_1 = 1$ (в нашей нумерации $p_2 = 2$, так как мы начали последовательность p_i с 1, не являющейся простым числом). Так праймориал $p_5\# = p_1 * p_2 * p_3 * p_4 * p_5 = 1 * 2 * 3 * 5 * 7 = 210$.

Запись числа в прайморадичной системе делится на позиции, разделённые двоеточиями :. Позиции нумеруются справа налево. Самая правая позиция имеет номер 1 и правее неё двоеточие не ставится. Вместо этого снизу приписывается знак $_\#$, помечающий, что это прайморадичная запись. На k-ой позиции допускается указывать десятичное число d_k, такое что $0 \leq d_k \leq p_{k+1} - 1$. На первой справа позиции может быть указан либо 0, либо 1. На второй справа позиции может стоять либо 0, либо 1, либо 2. На третьей справа позиции может быть либо 0, либо 1, либо 2, либо 3, либо 4. И так далее.

Допускается наличие незначащих нулей в левых позициях записи числа. Незначащим является любой нуль, находящийся левее первой встреченной при чтении слева направо ненулевой цифры, или, если записано нулевое число, то все нули, кроме самого правого. Чтобы понять, какое число записано, нужно выполнить вычисления: $d_n : d_{n-1} : \dots : d_i : \dots : d_2 : d_{1\#} = d_n * p_n\# + d_{n-1} * p_{n-1}\# + \dots + d_i * p_i\# + \dots + d_2 * p_2\# + d_1 * p_1\#$ Например, $6 : 3 : 0 : 1_\# = 6 * p_4\# + 3 * p_3\# + 0 * p_2\# + 1 * p_1\# = 6 * 30 + 3 * 6 + 0 * 2 + 1 * 1 = 180 + 18 + 0 + 1 = 199_{10}$. То же самое число может быть записано с незначащими нулями: $0 : 0 : 6 : 3 : 0 : 1_\#$. Заметим, что значащая позиция прайморадичной записи не содержит незначащих нулей, то есть, записывая d_k, не используют излишние нули слева.

Составьте программу, принимающую на вход в первой строке десятичное число N – положительное натуральное число ($1 \leq N \leq 4000$) – длину последовательности, а в последующих N строках – прайморадичные записи чисел X_i без завершающего знака $_\#$, $1 \leq i \leq N$. Известно, что в каждой записи числа X_i не более чем 25 позиций. Программа находит, количество тех чисел последовательности, которые равны наибольшему из введённых X_i. Программа выводит в единственной строке искомое количество вхождений максимума в последовательность, записанное десятичным натуральным числом без знака.

Формат ввода: В первой строке содержится десятичное число N — длина последовательности ($1 \leq N \leq 4000$). В следующих N строках содержатся прайморадичные записи чисел X_i без завершающего знака $_\#$, $1 \leq i \leq N$. В каждой такой записи не более чем 25 позиций, разделённых двоеточиями :. В каждой позиции неотрицательное целое десятичное число, допустимое правилами прайморадичной системы.

Формат вывода: Выводится беззнаковое десятичное натуральное число, равное количеству всех тех X_k, для которых верно, что $X_k = \max_{i=1 \dots N} X_i$.

```
Ввод примера №1:
1
4:0:0:0
Вывод примера №1:
1
Ввод примера №2:
3
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
0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:0
0
Вывод примера №2:
3
Ввод примера №3:
```

```
2
96:88:82:78:72:70:66:60:58:52:46:42:40:36:30:28:22:18:16:12:10:6:4:2:1
96:88:82:78:71:70:66:60:58:52:46:42:40:36:30:28:22:18:16:12:10:6:4:2:1
Вывод примера №3:
1
```

Решение задачи 2

В решении можно запрограммировать следующие подзадачи: 1) считывание очередного числа и представление его в виде строки из 50 символов с незначащими нулями, дополняющими запись прайморадичного числа до 25 позиций, а каждую считанную позицию записи – до двух цифр (в такой строке позиции идут подряд без разделения :); 2) подсчёт количества всех элементов последовательности, равных максимальному, основаный на посимвольном сравнении строк, эквивалентном поразрядному сравнению считанных чисел. Для решения второй подзадачи достаточно одного прохода по последовательности, в котором совмещены построчный ввод чисел и их обработка. Следует хранить текущий рекорд (максимальное среди всех чисел, которые программа успела считать) и количество всех элементов последовательности равных текущему рекорду. Очередное число после считывания сравнивается с рекордом. При равенстве количество max-ов увеличивается на 1. Если очередное число меньше, то делается переход к обработке следующего числа. Если очередное число больше, то оно становится рекордом, количество max-ов приравнивается 1. По окончании обработки выводится найденное количество max-ов.

Код возможного решения

```
program PRIMORADICS79 (input, output);
type    primoradics = array [1..50] of char;
        answer = record quantity : word; max : primoradics end;
var  CURNUM : primoradics; N, I : word; CHECK : integer; CURANSWER : answer;
procedure readnumber(var P : primoradics);
var     S : string; I, J, K, L : byte;
begin   readln(S); J:=50; I:=Length(S);
        while (I > 0) and (J > 0) do begin
          K := 2;
          while (I > 0) and (S[I] <> ':') and (J > 0) and (K > 0) do begin
            P[J] := S[I]; I := I - 1; J := J - 1; K := K - 1
          end;
          if (I > 0) and (S[I] = ':') and (J > 0) and (K > 0) then begin
            P[J] := '0'; J := J - 1; K := K - 1
          end;
          if (I > 0) and (S[I] = ':') then I := I - 1
        end;
        for I := J downto 1 do P[I] := '0'
end;
begin   readln(N);
        with CURANSWER do begin
            quantity := 1; readnumber(max);
            for I := 2 to N do begin
                readnumber(CURNUM);
                CHECK := CompareChar(max, CURNUM, 50);
                if (CHECK < 0) then begin quantity := 1; max := CURNUM end
                else if (CHECK = 0) then begin quantity := quantity + 1 end;
            end;
            writeln(quantity);
        end;
end.
```

Критерий оценивания решений задачи 2

Критерием является количество тестов, верно пройденных программой, составленной участником. Высчитывается доля (в процентах) таких тестов по отношению к общему количеству тестов. Полученное количество процентов, округлённое до целого, определяет количество технических баллов за решение.

Задача 3. Сундуки

Скупой рыцарь хранит в подвале множество сундуков. Сундуки занумерованы, начиная с единицы. В сундуках лежат монеты. О содержимом каждого сундука есть опись. В описи могут использоваться только следующие строчные латинские буквы: *a* означает монету номиналом *1 копейка*; *b – 5 копеек*; *c – 10 копеек*; *d – 50 копеек*; *e – 1 рубль или 100 копеек*; *f – 2 рубля*; *g – 5 рублей*; *h – 10 рублей*; *i – 25 рублей*. Буква входит в опись столько раз, сколько монет соответствующего номинала лежит в сундуке. Буквы в описи могут идти в произвольном порядке, так как скупой рыцарь добавляет монеты в сундуки без какой-либо системы и тут же дописывает нужные буквы в описи. Например, опись *abbahihihi* означает, что в сундуке лежат 105 рублей и 12 копеек (две монеты номиналом *1 копейка*; две монеты номиналом *5 копеек*; три монеты номиналом *10 рублей*; три монеты номиналом *25 рублей*). Опись пустого сундука является пустой строкой.

Скупой рыцарь хочет найти в своём подвале два сундука, таких, чтобы после суммирования денежных сумм, лежащих в них, получился максимальный результат. Если в подвале есть несколько искомых пар сундуков, то скупой рыцарь выбирает из них такую пару, что сумма номеров сундуков минимальна.

Составьте программу, принимающую на вход в первой строке десятичное число N – положительное натуральное число ($2 \le N \le 1000$) – количество сундуков, а в последующих N строках – описи сундуков с номерами от 1 до N. Известно, что в каждой описи не более чем 1000 букв, то есть, в каждом сундуке не более чем 1000 монет. Программа находит номера K и L такие, что $K < L$, суммарное количество денег в K-ом и L-ом сундуках наибольшее и сумма $K + L$ наименьшая. Программа выводит в первой строке найденное число K, а во второй строке – L. Каждое число записывается в десятичной системе без знака.

Формат ввода: В первой строке содержится десятичное число N — количество сундуков ($2 \le N \le 1000$). В следующих N строках содержатся описи монет из сундуков – последовательности, в которых могут встретиться только строчные латинские буквы *a*, *b*, . . . , *i*. Длины строк находятся в диапазоне от 0 до 1000 включительно.

Формат вывода: В первой строке выводится беззнаковое десятичное натуральное число K. Во второй строке выводится беззнаковое десятичное натуральное число L. Числа K и L таковы, что $K < L$, суммарное количество денег в K-ом и L-ом сундуках наибольшее и сумма $K + L$ наименьшая.

```
Ввод примера №1:
2
abbahihihi
cdffge
Вывод примера №1:
1
2
Ввод примера №2:
3
i
i
ai
Вывод примера №2:
1
3
Ввод примера №3:
4
bfffff
h
eeeeeaeeeee
hb
Вывод примера №3:
1
4
```

Решение задачи 3

В решении можно запрограммировать следующие подзадачи: 1) считывание очередной описи сундука и представление её в виде массива из 9 элементов, в каждом из которых хранится количество монет соответствующего номинала); 2) нормализацию описи в виде массива, то есть приведение к виду, когда монет номиналом 1 копейка не более чем 4 (остальные конвертируются в 5-тикопеечные монеты), монет номиналом 5 копеек не более 1 (остальные конвертируются в 10-тикопеечные монеты), монет номиналом 10 копеек не более 4 (остальные конвертируются в 50-тикопеечные монеты), монет номиналом 50 копеек не более чем 1 (остальные конвертируются в 1-норублёвые монеты), монет номиналом 1 рубль не более 1 (остальные конвертируются в 2-хрублёвые монеты), монет номиналом 2 рубля не более чем 2 (остальные конвертируются в 5-тирублёвые монеты, при этом либо используется ещё одна 1-норублёвая монета, либо 1 рубль остатка остаётся и количество 1-норублёвых монет прирастает), монет номиналом 5 рублей не более чем 1 (остальные конвертируются в 10-тирублёвые монеты), монет номиналом 10 рублей не более чем 2 (остальные конвертируются в 25-тирублёвые монеты, при этом либо используется ещё одна 5-тирублёвая монета, либо 5 рублей остатка остаётся и количество 5-тирублёвых монет прирастает); 3) поэлементное сравнение двух нормализованных описей в виде массивов; 4) поиск двух наибольших элементов в последовательности описей и вывод их номеров. Для решения достаточно одного прохода по последовательности, в котором совмещены посимвольный ввод описей и их обработка. Следует хранить текущий рекорд (максимум среди всех описей, которые программа успела считать) и текущий «почти рекорд» (наибольшую опись среди остальных описей, не превышающую текущий рекорд). Очередная опись после считывания сравнивается с рекордом. Если рекорд не побит, то она сравнивается с «почти рекордом». Если очередная опись меньше «почти рекорда», то делается переход к обработке следующей описи. Если очередная опись больше «почти рекорда», то она становится «почти рекордом». Если очередная опись больше рекорда, то она становится рекордом, а прежний рекорд становится «почти рекордом». По окончании обработки выводятся номера рекорда и «почти рекорда» по возрастанию.

Код возможного решения

```
program TREASURECHESTS79 (input, output);
type    chests = array ['a'..'i'] of word;
        answer = record number : word; chest : chests end;
var  CURCHEST : chests; N, I : word; CHECK : integer; CURMAX, CURPREDMAX : answer;
procedure readchest(var CHEST : chests);
var     CH, J : char; I : word;
begin
   for J := 'a' to 'i' do CHEST[J] := 0;
   if (not EOLn) then begin
        read(CH);
        I:=999;
        while (I > 0) and (not EOLn) do begin
          CHEST[CH] := CHEST[CH] + 1;
          read(CH);
          I := I - 1;
        end;
        if (EOLn) then CHEST[CH] := CHEST[CH] + 1;
        (* for J := 'i' downto 'a' do write(CHEST[J], ' '); writeln; *)
        for J := 'a' to 'h' do
          case J of
            'a', 'c': if (CHEST[J] > 4) then begin
                        CHEST[succ(J)] := CHEST[succ(J)] + CHEST[J] div 5;
                        CHEST[J] := CHEST[J] mod 5
                      end;
            'b', 'd', 'e', 'g': if (CHEST[J] > 1) then begin
                        CHEST[succ(J)] := CHEST[succ(J)] + CHEST[J] div 2;
                        CHEST[J] := CHEST[J] mod 2
                      end;
            'f', 'h': if (CHEST[J] > 2) then begin
```

```
                    CHEST[succ(J)] := CHEST[succ(J)] + (CHEST[J] * 2) div 5;
                    I := (CHEST[J] * 2) mod 5;
                    if odd(I) then
                       if (CHEST[pred(J)] > 0) then begin
                          CHEST[pred(J)] := CHEST[pred(J)] - 1;
                          I := I + 1 end
                       else begin
                          CHEST[pred(J)] := CHEST[pred(J)] + 1;
                          I := I - 1 end;
                    CHEST[J] := I div 2;
                 end;
          end;
          (* for J := 'i' downto 'a' do write(CHEST[J], ' '); writeln *)
   end
end;
function CompareChest(CHEST1, CHEST2: chests) : integer;
var J : char; RESULT : integer;
begin   RESULT := 0;
        J := 'i';
        while (RESULT = 0) and (J >= 'a') do begin
            if (CHEST1[J] > CHEST2[J]) then
                RESULT := 1
            else if (CHEST1[J] < CHEST2[J]) then
                RESULT := -1;
            J := pred(J)
        end;
        CompareChest := RESULT
end;
begin   readln(N);
        with CURMAX do begin
            number := 1;
            readchest(chest);
            readln;
        end;
        readchest(CURCHEST);
        CHECK := CompareChest(CURMAX.chest, CURCHEST);
        if (CHECK < 0) then begin
            CURPREDMAX := CURMAX;
            with CURMAX do begin
               number := 2;
               chest := CURCHEST
            end end
        else
            with CURPREDMAX do begin
               number := 2;
               chest := CURCHEST
            end;
        for I := 3 to N do begin
            readln;
            readchest(CURCHEST);
            CHECK := CompareChest(CURMAX.chest, CURCHEST);
            if (CHECK < 0) then begin
                CURPREDMAX := CURMAX;
                with CURMAX do begin
                   number := I;
                   chest := CURCHEST
```

```
                end end
            else begin
                CHECK := CompareChest(CURPREDMAX.chest, CURCHEST);
                if (CHECK < 0) then
                    with CURPREDMAX do begin
                       number := I;
                       chest := CURCHEST
                end
            end
        end;
        if (CURMAX.number > CURPREDMAX.number) then begin
            writeln(CURPREDMAX.number);
            writeln(CURMAX.number) end
        else begin
            writeln(CURMAX.number);
            writeln(CURPREDMAX.number)
        end
end.
```

Критерий оценивания решений задачи 3

Критерием является количество тестов, верно пройденных программой, составленной участником. Высчитывается доля (в процентах) таких тестов по отношению к общему количеству тестов. Полученное количество процентов, округлённое до целого, определяет количество технических баллов за решение.

Задача 4. Измерения температуры

В результате измерения были получены среднедневные температуры за N последовательных дней ($1 \leq N \leq 10^7$). Иннокентий решил найти максимальную температуру на всех последовательных интервалах длины K ($1 \leq K \leq 10^4$, $K \leq N$). То есть, на отрезках $[0..K-1]$, $[1..K]$, $[2..K+1]$ и т. д.. Результаты поисков не понравились Иннокентию, поэтому он решил расширить слева и справа каждый i-й отрезок на l_i, r_i соответственно. Считайте, что если при этом происходит выход за границы исходной последовательности, то числа там $-inf$. На стандартном потоке вводится число N и N чисел, задающих последовательность измерений. Затем следует K – длина отрезка поиска, затем следуют $N-K+1$ пары положительных чисел, не больших 1000 и не больших K, – l_i, r_i соответственно.

Для каждого из $N-K+1$ отрезков длины K с соответствующими расширениями выведите максимум на них.

Пример:

```
Ввод:
10
1 2 3 1 3 7 8 5 3 1
3
0 1
1 1
0 1
0 0
0 0
0 0
0 0
1 0
Вывод:
3
3
7
7
8
8
8
8
```

Код возможного решения задачи 4

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

#ifdef _WIN32
#define getchar_unlocked() _getchar_nolock()
#define _CRT_DISABLE_PERFCRIT_LOCKS
#endif

using namespace std;

int a[10000000];
int L[10000000];
int R[10000000];

char buf_in[10000000];
char buf_out[10000000];

void fastscan(int &number)
{
    bool negative = false;
    int c = getchar_unlocked();
```

```
    number = 0;
    if (c=='-')
    {
        negative = true;
        c = getchar_unlocked();
    }

    for (; (c>47 && c<58); c=getchar_unlocked())
        number = number *10 + c - 48;

    if (negative)
        number *= -1;
}

void fastscan_(int &number)
{
    int c = getchar_unlocked();
    number = 0;
    for (; (c>47 && c<58); c=getchar_unlocked())
        number = number *10 + c - 48;
}

int main()
{
    setvbuf ( stdout , buf_out , _IOFBF , sizeof(buf_out) );
    setvbuf ( stdin , buf_in , _IOFBF , sizeof(buf_in) );
    std::ios_base::sync_with_stdio(false);
    int N;
    int K;
    fastscan_(N);

    for (int i = 0; i < N; ++i) {
        fastscan(a[i]);
    }
    fastscan_(K);

    for (int i = 0; i < N; ++i) {
        if (i % K == 0) {
            L[i] = a[i];
        } else {
            L[i] = max(L[i - 1], a[i]);
        }
    }


    for (int i = N - 1; i >= 0; --i) {
        if (i == N - 1 || (i + 1) % K == 0) {
            R[i] = a[i];
        } else {
            R[i] = max(a[i], R[i + 1]);
        }
    }

    for (int i = 0; i < N - K + 1; ++i) {
        int l, r;
        fastscan_(l);
```

```
        fastscan_(r);
        int tl = max(0, i - l);
        int tr = min(N - 1, i + K - 1 + r);
        int res = max(L[tr], R[tl]);
        if ((tl / K + 1) * K < tr / K * K) {
            res = max(res, R[(tl / K + 1) * K]);
        }
        if ((tl / K + 2) * K < tr / K * K) {
            res = max(res, R[(tl / K + 2) * K]);
        }
        printf("%d ", res);
    }
    return 0;
}
```

Критерий оценивания решений задачи 4

Критерием является количество тестов, верно пройденных программой, составленной участником. Высчитывается доля (в процентах) таких тестов по отношению к общему количеству тестов. Полученное количество процентов, округлённое до целого, определяет количество технических баллов за решение.

Задача 5. Римский Бармаглот

Компания Barmaley's Computing выпустила новый электронный вычислитель - Barmaglot-2. Поскольку у сотрудников компании несколько странные представления о технологиях, компьютер остался довольно непривычным.

Во-первых, исходные данные он читает с ленты, где могут быть записаны целые числа и латинские буквы. Чтобы показать, что ввод данных закончен, ленту нужно просто оторвать.

Во-вторых, результаты вычислений компьютер печатает на точно такой же ленте. Центральный процессор компьютера оснащён одним регистром; вместо оперативной памяти компьютер оснащён очередью и стеком неограниченного размера. Регистр, а также каждая ячейка очереди и стека могут содержать либо число, либо латинскую букву или пробел, либо специальное значение [BARMALEY], которое используется для обозначения логической лжи, при том что любое другое обозначает истину.

Программа для Barmaglot'а представляет собой строку символов, каждый из которых задаёт машинную команду; программа, состоящая из символов-команд, выполняется последовательно слева направо, кроме двух команд, которые могут нарушить эту последовательность.

Команды a, b, c и все остальные латинские буквы означают «занести данную букву в регистр».

Команды 0, 1, ..., 9 означают «занести в регистр соответствующее число».

Команда @ означает занести в регистр пробел.

Команды +, -, *, /, % означают соответствующие арифметические действия, операции целочисленные. % - операция взятия остатка

Команды <, >, = означают сравнение двух чисел или двух символов, & и | означают логическое «и» и логическое «или»;

Все эти команды используют значение из регистра в качестве левого операнда, значение с вершины стека в качестве правого (оно при этом из стека извлекается), результат заносится обратно в регистр.

Команда # умножает содержимое регистра в десять раз,

Команда _ делит содержимое регистра в десять раз.

Команда работает как логическое отрицание содержимого регистра: если там значение [BARMALEY], то заносится значение 1, если любое другое - заносится значение [BARMALEY].

Команда . выдаёт текущее значение регистра на печать

Команда ? вводит очередное число(целиком, а не по разрядам), а если на вводе кончилась (оборвалась) лента, заносит в регистра значение [BARMALEY].

Команда] заносит значение из регистра в стек;

Команда [извлекает значение с вершины стека и заносит его в регистра (если извлекать нечего, в регистр заносится [BARMALEY]);

Команда ~ меняет местами значения в регистра и на вершине стека.

Команда } заносит значение из регистра в очередь,

Команда { извлекает из очереди самое старое значение и помещает в аккумулятор (или помещает туда [BARMALEY], если очередь пуста).

Команды (и) предназначены для организации ветвлений и циклов и всегда должны в программе стоять парами, то есть в программе должен обязательно соблюдаться баланс круглых скобок. Выполняются они так. Команда (, если в регистре [BARMALEY], идёт по программе вперёд, пока не найдёт парную скобку, и после этого выполнение продолжится со следующей за этой закрывающей скобкой позиции; если в регистре было что-то другое, команда вообще ничего не делает, то есть выполнение продолжается прямо с команды, следующей за ней. Команда), наоборот, если в регистре [BARMALEY], не делает ничего, тогда как если там что-то другое, просматривает программу назад до парной круглой скобки, после чего продолжает выполнение с команды, стоящей после такой скобки справа.

Наконец, команда ïпрекращает выполнение программы, при этом выполнение считается успешным. Если программа кончилась, не встретив эту команду, она завершается аварийно.

Пробелы в программе игнорируются.

Пример программы, которая печатает традиционную строчку "HELLO WORLD":

H.E.L.L.O.@.W.O.R.L.D."

В вашем распоряжении оказался эмулятор данного компьютера (http://ejudge.cs.msu.ru/barmaglot2/) и требуется написать для него программу, которая переведет введенное число X в десятичной

системе счисления ($1 \leq X \leq 100$) в римскую систему счисления. Число в римской системе счисления записывается заглавными латинскими буквами.

Пример:

```
Ввод:
38
Вывод:
XXXVIII
```

Код возможного решения задачи 5

```
?]]_]]]]]]]]]]_]1=(C.!)9=(X.C.!)8=(L.X.X.X.!)7=(L.X.X.!)6=(L.X.!)5=(L.!)4=(X.L.!)3=(X.X.X.!)2
=(X.X.!)1=(X.!)[#~-]]]]]]]]]]9=(I.X.!)8=(V.I.I.I.!)7=(V.I.I.!)6=(V.I.!)5=(V.!)4=(I.V.!)3
=(I.I.I.!)2=(I.I.!)1=(I.!)"
```

Критерий оценивания решений задачи 5

Критерием является количество тестов, верно пройденных программой, составленной участником. Высчитывается доля (в процентах) таких тестов по отношению к общему количеству тестов. Полученное количество процентов, округлённое до целого, определяет количество технических баллов за решение.

Задача 6. Микрокалькулятор

В рамках импортозамещения вам предстоит освоить программирование на микрокалькуляторе МК-61. Эмулятор микрокалькулятора доступен здесь:
https://sergeanvarov.github.io/russian/mk61/Эмулятор%20МК-61.html

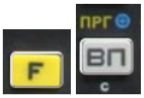
Руководство по эксплуатации доступно здесь:
https://www.wass.net/manuals/Elektronika%20MK-61.pdf

В качестве примера использования рассмотрим написание программы, которая вычисляет гипотенузу прямоугольного треугольника по двум катетам. Предположим, что перед началом вычисления длины катетов записываются на верхушке стека, то есть в регистрах X и Y.

Последовательность шагов для программирования:

1. Включите калькулятор рычажком 

2. Перейдите на нулевой адрес программы 

3. Перейдите в режим программирования 

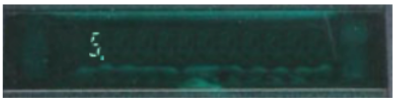
4. Программа:  (14),  (22),  (14),  (22),  (10),  (21),  (50).

5. Перейдите в режим счета: 

6. Перейдите на нулевой адрес программы: 

7. Занесите два числа в стек: 

8. Запустите программу: 

9. На дисплее будет отображаться ответ: 

Можно скопировать введенную программу в окно ввода с помощью кнопки "прочитать программу из памяти". В окне ввода будет следующее:

```
<-> Fx^2 <-> Fx^2 + F√ С/П 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    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   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    0   0    0 0  0   0 0 0
0   0    0   0    0 0  0   0 0 0
0   0    0   0    0
```

Именно содержимое этого окна нужно будет сдать в тестирующую систему в качестве решения следующей задачи. Для калькулятора напишите программу, которая переводит число от 1 до 100, записанное в десятичной записи, в римскую запись. Поскольку калькулятор не позволяет отображать буквы, буквы в римской записи числа будут кодироваться цифрами следующим образом:

I кодируется как 1
V кодируется как 2
X кодируется как 3
L кодируется как 4
C кодируется как 5

Таким образом, число 38 представляется в закодированной римской записи числом 3332111.

На проверку сдайте программу, скопированную из окна ввода после нажатия кнопки «прочитать программу из памяти», которая выполняет перевод из десятичной записи в римскую.

Пример:

```
Ввод:
38
Вывод:
3332111
```

Код возможного решения задачи 6

```
хП1  0    хП2 Пх2  +    1    0    ×    хП2 Пх1
2    F10ˣ -   Fx≥0 19   хП1  5    БП   03  Пх1
9    0    -   Fx≥0 30   2    F10ˣ +   БП  57
Пх1  5    0   -    Fx≥0 40   хП1  4   БП  03
Пх1  4    0   -    Fx≥0 51   5    0   +   БП
57   Пх1  1   0    -    Fx≥0 61   хП1 3   БП
03   Пх1  9   -    Fx≥0 71   1    0   +   БП
А0   Пх1  5   -    Fx≥0 80   хП1  2   БП  03
Пх1  4    -   Fx≥0 89   5    +    БП  А0  Пх1
Fx=0 97   Пх2 1    0    ÷    С/П  Пх1 1   -
хП1  1    БП  03   0
```

Критерий оценивания решений задачи 6

Критерием является количество тестов, верно пройденных программой, составленной участником, а также свойство этой программы, заключающееся в том, требуется ли выключение и повторное включение исполнителя после выполнения им программы. Высчитывается доля (в процентах) верно пройденных тестов по отношению к общему количеству тестов. Полученное количество процентов, округлённое до целого, определяет начальное количество технических баллов за решение. Из начального количества технических баллов вычитается штраф, 10 баллов, если требуется перезапуск исполнителя после выполнения программы. Если перезапуск не требуется, то штраф не вычитается. Например, решение может быть оценено в 80 технических баллов в следующих случаях: либо решение проходит верно 80% тестов и не требуется перезапуск исполнителя, либо решение проходит 90% тестов и требуется перезапуск исполнителя.

Перевод технических баллов в оценку

По каждой задаче определялись самые удачные её решения, отправленные участником. Технические баллы за них суммировались. Высчитывалась доля (в процентах), которую составила полученная сумма по отношению к 440 техническим баллам. Полученное количество процентов, округлённое до целого, становилось оценкой.