Управление по делам образования г. Челябинска
Управление информатизации образования ЧелГУ
Центр информатизации образования лицея № 11

ГОРОДСКАЯ ОЛИМПИАДА

ШКОЛЬНИКОВ 5–8 КЛАССОВ

ПО ПРОГРАММИРОВАНИЮ

2001–2002 ГГ.

Челябинск 2003

Управление по делам образования г. Челябинска
Управление информатизации образования ЧелГУ
Центр информатизации образования лицея № 11

ГОРОДСКАЯ ОЛИМПИАДА

ШКОЛЬНИКОВ 5–8 КЛАССОВ

ПО ПРОГРАММИРОВАНИЮ

2001–2002 ГГ.

Челябинск 2003

ББК 3973
Г 701

Г 701 Городская олимпиада школьников 5–8 классов по программированию 2001–2002 гг. / Под ред. М.Л. Цымблера, К.Р. Овчинниковой, И.В. Курбатовой. Челябинск: Челяб. гос. ун-т, 2003. 34 с.

ISBN 5–7271–0616–8

Издание содержит учебно-методические и информационные материалы, связанные с проведением олимпиады по программированию среди школьников 5–8 классов г. Челябинска. Рассмотрены правила проведения олимпиады. Даны технические результаты олимпиад 2001–2002 гг. Приведены олимпиадные задачи и указания по их решению.

Печатается по решению оргкомитета олимпиады.

Рецензент А.Н. Янченко, канд. техн. наук, доцент

Г $\frac{2405000000 - 009}{4К\ 8(03) - 03}$ Без объявл. ББК 3973.2 – 018.1м

ISBN 5–7271–0616–8

1. Городские олимпиады по программированию среди школьников

В целях стимулирования интереса школьников к углубленному изучению программирования и информатики и выявления одаренных школьников в области программирования и информатики в 2001 г. в Челябинске была проведена первая *городская олимпиада по программированию для школьников 5–8 классов*.

Организаторами олимпиады стали Управление по делам образования, Челябинский государственный университет и лицей № 11 г. Челябинска. Данные организации осуществляли общее руководство и финансирование олимпиады. Олимпиада проводилась на базе лицея № 11 и Челябинского государственного университета. Специальное программное обеспечение, необходимое для проведения олимпиады, на безвозмездной основе предоставил Челябинский государственный университет.

Учитывая интерес, проявленный учителями и школьниками к соревнованиям, организаторы приняли решение *проводить олимпиаду ежегодно*. **Во втором разделе** данной брошюры приводятся *технические результаты I и II олимпиад*.

Схема проведения олимпиады схожа с правилами Командного чемпионата мира по программированию среди студентов, который проводится международной организацией Association for Computing Machinery (ACM). В настоящее время ACM является наиболее авторитетным сообществом специалистов в области компьютеров, объединяя более 80 тыс. ученых, инженеров, программистов и студентов. ACM проводит многочисленные научные конференции по основным проблемам информатики, а также ежегодно присуждает премию Тьюринга, которую называют нобелевской премией в области компьютеров. С 1977 г. ACM проводит Командный чемпионат мира по программированию среди вузов. За более чем двадцатилетнюю историю сформировались правила этих соревнований. Участникам соревнований предлагается в течение определенного времени решить максимальное число задач. Победитель определяется по наибольшему числу решенных задач. В случае равенства числа решенных задач выигрывает тот, кто затратил на их решение меньше времени. **В третьем разделе** данной брошюры детально рассмотрены правила проведения олимпиады.

Отличия олимпиад по программированию и информатике *от олимпиад по другим учебным дисциплинам* заключаются в предлагаемых участникам задачах и способе проверки решений. *Задача* направлена на составление программы, которая будет выдавать верные выходные данные для входных данных, подготовленных жюри и заранее неизвестных участнику. При проведении соревнований используется *система автоматизированной проверки решений задач*. **В четвертом разделе** данной брошюры приведены задачи I и II олимпиад и указания по их решению.

В пятом разделе брошюры находится *Положение об олимпиаде*, в котором описаны порядок организации олимпиады, финансовое и методическое обеспечение, персональный состав оргкомитета и др.

Оргкомитет олимпиады надеется, что городская олимпиада по программированию среди школьников 5–8 классов будет способствовать выявлению одаренных школьников в области программирования и информатики. А опыт, приобретенный молодыми дарованиями, поможет им в выступлениях на олимпиадах по программированию и информатике областного и российского уровней.

2. Результаты олимпиады

В олимпиаде 2001 г. приняли участие следующие образовательные учреждения города: лицеи № 11 и 31 (Управление по делам образования г. Челябинска), школы № 50 Калининского района, № 76 и 77 Ленинского района, № 93 Курчатовского района, а также Центр детского творчества Металлургического района.

В личном зачете в параллели 5–6 классов победили:

1. Зубов Максим, лицей № 11, 6 класс (решил 4 задачи из 5).
2. Полетаев Константин, лицей № 11, 6 класс (3 задачи из 5).

В личном зачете в параллели 7–8 классов победили:

1. Мандриков Евгений, лицей № 31, 8 класс (решил 5 задач из 5; 241 мин. штрафного времени).
2. Немкин Никита, лицей № 11, 8 класс (5 задач из 5; 401 мин.)
3. Денисов Антон, школа № 77, 8 класс (5 задач из 5; 412 мин.).

В командном зачете места распределились следующим образом:

1. Лицей № 11 (руководитель команды Глушкова Ольга Анатольевна).
2. Лицей № 31 (руководитель команды Погодин Александр Петрович).
3. Школа № 77 (руководитель команды Абросимова Татьяна Юрьевна).

В олимпиаде 2002 г. приняли участие следующие образовательные учреждения города: лицеи № 11 и 31 (Управление по делам образования г. Челябинска), школы № 18 Тракторозаводского района, № 35 и 89 Курчатовского района.

Поскольку ни одна команда не выставила для участия учеников 5–6 классов, в личном зачете итоги подводились только *в параллели 7–8 классов*. Победили:

1. Ермолаев Иван, лицей № 31, 8 класс (решил 3 задачи из 5).
2. Стерликов Петр, лицей № 11, 8 класс (2 задачи из 5; 313 мин. штрафного времени).
3. Илюшенков Михаил, лицей № 31, 8 класс (2 задачи из 5; 630 мин.).

В командном зачете места распределились следующим образом:

1. Лицей № 31 (руководитель команды Погодин Александр Петрович).
2. Лицей № 11 (руководитель команды Яковлева Татьяна Геннадьевна).
3. Школа № 18 (руководитель команды Костромцов Алексей Сергеевич).

3. Правила соревнований

Городская олимпиада школьников 5–8 классов проводится по схеме студенческих чемпионатов мира по программированию.

Олимпиада длится *три часа*. Участнику предоставляется *один компьютер* и предлагается к решению *пять задач*.

Характер задач предполагает, что участники продемонстрируют мастерство как в алгоритмизации задач, так и в составлении эффективных программ, реализующих выбранные для решения алгоритмы.

Задачи имеют различную *сложность*. В комплект задач включаются "утешительная" задача (которая может быть решена всеми участниками), задачи невысокой и средней сложности, а также задача повышенной сложности (которая, возможно, не будет решена ни одним из участников). *Подбор комплекта задач* осуществляется с таким расчетом, чтобы ни один из участников (возможно, за исключением победителя) не смог решить все задачи – в силу их сложности и малого количества времени на их решение.

Задача включает в себя формулировку условия, описание формата входных данных, описание формата выходных данных и примеры входных и соответствующих им выходных данных. *Входные данные* задачи необходимо прочитать из текстового файла INPUT.TXT. *Выходные данные* задачи необходимо записать в текстовый файл OUTPUT.TXT.

Решением задачи является файл с исходным текстом программы на одном из следующих языков программирования: Pascal, BASIC, C, C++. Участник может решать различные задачи на различных языках программирования. При решении задач не разрешается использовать книги, справочники, руководства, электронные словари, листинги программ и т.п.

Отсылка решения на проверку жюри и *просмотр текущих результатов* соревнований осуществляется с помощью специального программного обеспечения. Для сохранения интриги обновление текущих результатов соревнований прекращается за полчаса до окончания соревнований.

Проверка решений задач проводится во время соревнований с помощью автоматизированной проверяющей системы. Жюри определяет, что *задача решена*, если решение выдает правильные ответы на *каждом* из *эталонных тестов* данной задачи, подготовленных жюри и *не известных* участникам соревнований. В случае, если решение выдает неверный ответ хотя бы на одном из эталонных тестов, задача не считается решенной и участнику сообщается только порядковый номер эталонного теста, на котором его решение выдало неверный ответ.

Определение победителей осуществляется следующим образом. Побеждает участник, решивший наибольшее число задач. В случае равенства числа решенных задач побеждает участник, имеющий наименьшее штрафное время. *Штрафное время* определяется как суммарное время решения задач, прошедшее с начала соревнований с добавлением 20 минут за каждую неудачную попытку сдать задачу. Штрафное время подсчитывается только для решенных задач. Например, если задача была решена на 70 минуте соревнований с 3 попытки, то штрафное время составит 70+3*20=130 минут.

Отображение текущих результатов соревнований осуществляется в форме таблицы. Рассмотрим следующий пример таблицы соревнований.

Место	Участник	Задачи					Решено	Штрафное время
		A	B	C	D	E		
1	Иванов	+	+1	+2	+3	+1	5	546
2	Петров	+	+2	+	+4	.	4	217
3	Сидоров	–6	+	+	+1	.	3	189
4	Егоров	+	+2	–1	+	–2	3	371

Здесь используются следующие условные обозначения. Задачи обозначены начальными буквами латинского алфавита. Символ "плюс" означает, что соответствующая задача решена. Символ "минус" – соответствующая задача не решена. Число показывает количество неудачных попыток сдать решение задачи.

Например, участник Петров решил задачи A, B, C и D, а решение задачи E не пытался сдать жюри. При этом решения задач A и C Петров сдал с первой попытки, решение задачи B – с 3 попытки, а решение задачи D – с 5 попытки. Участник Сидоров сделал 6 неудачных попыток сдать решение задачи A (и в итоге не решил ее), решил одинаковое с участником Егоровым количество задач, но занял более высокое место, так как набрал меньше штрафного времени.

4. Разбор задач олимпиады

Ниже приводятся тексты задач олимпиад 2001 и 2002 гг., а также указания к их решению и эталонные тесты, которые использовались для проверки решений участников. Авторы не утверждают, что приведенные решения являются единственно верными, а тесты к задачам – наиболее полными.

Задачи олимпиады 2001 г.

Задача "Сумма чисел" (пробный тур)

Напишите программу, которая находит сумму заданных целых чисел.

Входные данные

В первой строке файла входных данных записано одно целое число ***N*** – количество чисел, сумму которых требуется подсчитать, 1<=***N***<=1000. Далее идет ***N*** строк, в каждой из которых записано одно целое число, не превосходящее 1000.

Выходные данные

В файл выходных данных требуется записать одно число, представляющее сумму заданных чисел.

Примеры входных и выходных данных

INPUT.TXT	OUTPUT.TXT
2 1 2	3
5 10 12 30 4 4	60

Указания к решению

Задача содержит лишь один "подводный камень". Поскольку сумма целых чисел в файле входных данных может превысить значение MaxInt (верхняя граница диапазона типа Integer, равна 32 767), то переменная, в которой будет храниться сумма чисел, должна иметь тип LongInt.

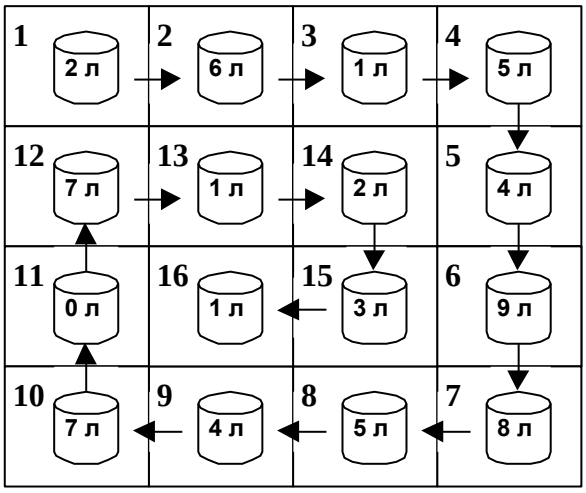
Эталонные тесты

№ п/п	INPUT.TXT	OUTPUT.TXT
1	2 1 2	3
2	5 10 12 30 4 4	60

№ п/п	INPUT.TXT	OUTPUT.TXT
3	34 1000 1000 1000 ... 1000	34000
4	34 -1000 -1000 -1000 ... -1000	-34000

Задача "Дом Винни-Пуха" (пробный тур)

Винни-Пух живет в доме квадратной формы. В каждой комнате дома хранится бочонок меда. Готовясь к зиме, Винни-Пух делает обход своего дома. Он заходит в каждую комнату, проверяет, сколько литров меда осталось в бочонке, и записывает результат в свою амбарную книгу. План дома и порядок обхода его комнат изображен на рисунке. В данном случае на одной стороне дома 4 комнаты, Винни-Пух двигается из комнаты № 1 в комнату № 16 по стрелкам.



Напишите программу, которая помогает Винни-Пуху сделать правильную запись в его амбарной книге.

Входные данные

В первой строке файла входных данных записано одно целое число N – число комнат на одной стороне дома Винни-Пуха, $1 <= N <= 10$. Далее идет план дома. Это N строк, в каждой из которых записано N целых неотрицательных чисел, не превосходящих 100. Число показывает количество литров меда в бочонке, который находится в соответствующей комнате дома.

Выходные данные

В файл выходных данных требуется поместить запись об обходе комнат дома. Для каждой посещенной комнаты в отдельной строке следует записать количество литров меда в бочонке, который находится в этой комнате.

Примеры входных и выходных данных

INPUT.TXT	OUTPUT.TXT
2 1 2 3 4	1 2 4 3
4 2 6 1 5 7 1 2 4 0 1 3 9 7 4 5 8	2 6 1 5 4 9 8 5 4 7 0 7 1 2 3 1

Указания к решению

Задача требует организации обхода матрицы исходных данных "по спирали" и сводится к аккуратному программированию вложенных циклов. При проверке решения следует обратить внимание на граничный случай, когда матрица исходных данных имеет размер 1×1. Фрагмент решения:

```
const
        MaxN = 10;
type
        TMatr = array [1..MaxN, 1..MaxN] of Integer;
var
        F: Text;
        i, j, N: Integer;
        A: TMatr;
begin
        …
        for i := 1 to N do begin
                for j := i to N–i+1 do
                        Write(F, A[i, j], ' ');
                for j := i+1 to N–i+1 do
                        Write(F, A[j, N–i+1], ' ');
                for j := N–i downto i do
                        Write(F, A[N–i+1, j], ' ');
                for j := N–i downto i+1 do
                        Write(F, A[j, i], ' ');
        end;
        …
end.
```

Эталонные тесты

№ п/п	INPUT.TXT	OUTPUT.TXT
1	2 1 2 3 4	1 2 4 3
2	4 2 6 1 5 7 1 2 4 0 1 3 9 7 4 5 8	2 6 1 5 4 9 8 5 4 7 0 7 1 2 3 1
3	4 1 2 3 4 12 13 14 5 11 16 15 6 10 9 8 7	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
4	1 1	1

Задача "Робинзон"

Робинзон, будучи на необитаемом острове, считал прожитые дни. Когда корабль забрал Робинзона с необитаемого острова, Капитан корабля должен был записать в судовой журнал, сколько полных лет, месяцев и дней прожил Робинзон на острове. Причем Капитан захотел сделать запись в журнале на русском языке, правильно согласуя числительные и слова "год", "месяц", "день" без ошибок в падежах. При этом Капитан считал, что в месяце 30 дней, а в году 12 месяцев (т.е. в году 360 дней).

Напишите программу, которая поможет Капитану сделать правильную запись в судовом журнале.

Входные данные

В файле входных данных записано одно целое число D – число дней, которые прожил Робинзон на необитаемом острове, $1 <= D <= 32000$.

Выходные данные

В файл выходных данных требуется поместить запись о том, сколько полных лет, месяцев и дней прожил Робинзон на острове. При этом нужно соблюдать следующие правила:

1. Слова "год", "месяц", "день" нужно согласовать с числительными. Например: "1 год", но "5 лет", "1 месяц", но "3 месяца", "1 день", но "23 дня".
2. Вывод числа полных лет, месяцев и дней нужно производить в отдельной строке. Если число полных лет, месяцев или дней, прожитых Робинзоном на острове, равно нулю, то соответствующую строку выводить не нужно.
3. Слова "год", "месяц", "день" нужно выводить БОЛЬШИМИ буквами.

Примеры входных и выходных данных

INPUT.TXT	OUTPUT.TXT
393	1 ГОД 1 МЕСЯЦ 3 ДНЯ
1800	5 ЛЕТ
1050	2 ГОДА 11 МЕСЯЦЕВ
10	10 ДНЕЙ
91	3 МЕСЯЦА 1 ДЕНЬ

Указания к решению

Для решения данной задачи требуется сначала с помощью операций целочисленного деления и взятия остатка найти число полных лет, месяцев и дней, проведенных Робинзоном на острове. Затем с помощью операции целочисленного взятия остатка и условного оператора найти нужную словоформу слов "год", "месяц", "день". Падеж этих слов зависит от последней цифры в числе лет, месяцев и дней. Фрагмент решения:

```
var
        F: Text;
        D: LongInt;
        Year, Month, Day: Integer;
begin
        …
        Year := D div 360;
        Month := D mod 360 div 30;
        Day := D mod 360 mod 30;
        if Year > 0 then begin
                Write(F, Year);
                if (Year mod 10 = 1) and (Year <> 11) then
```

```
                WriteLn(F, ' ГОД')
            else
                if (Year mod 10 in [1..5]) and ((Year < 10) or (Year > 20)) then
                    WriteLn(F, ' ГОДА')
                else
                    WriteLn(F, ' ЛЕТ');
        end;
        if Month > 0 then begin
            Write(F, Month);
            if Month = 1 then
                WriteLn(F, ' МЕСЯЦ')
            else
                if (Month > 1) and (Month < 5) then
                    WriteLn(F, ' МЕСЯЦА')
                else
                    WriteLn(F, ' МЕСЯЦЕВ');
        end;
        if Day > 0 then begin
            Write(F, Day);
            if (Day mod 10 = 1) and (Day <> 11) then
                WriteLn(F, ' ДЕНЬ')
            else
                if (Day mod 10 in [1..5]) and ((Day<10) or (Day>20)) then
                    WriteLn(F, ' ДНЯ')
                else
                    WriteLn(F, ' ДНЕЙ');
        end;
…
end.
```

Эталонные тесты

№ п/п	INPUT.TXT	OUTPUT.TXT
1	393	1 ГОД 1 МЕСЯЦ 3 ДНЯ
2	1800	5 ЛЕТ
3	1050	2 ГОДА 11 МЕСЯЦЕВ
4	10	10 ДНЕЙ
5	91	3 МЕСЯЦА 1 ДЕНЬ
6	32000	88 ЛЕТ 10 МЕСЯЦЕВ 20 ДНЕЙ
7	64000	177 ЛЕТ 9 МЕСЯЦЕВ 10 ДНЕЙ
8	4000	11 ЛЕТ 1 МЕСЯЦ 10 ДНЕЙ
9	4661	12 ЛЕТ 11 МЕСЯЦЕВ 11 ДНЕЙ

Задача "Чудо–гусли"

Плыл как–то купец Садко на своем корабле, вез товары на продажу да на гуслях играл. Услыхал ту игру Морской Царь, да так сильно она ему понравилась, что решил Морской Царь себе такие же гусли сладить. Остановил корабль Морской Царь и говорит купцу: "Помоги мне, купец Садко, такие гусли сладить. Есть у меня струны, но не знаю я

хитрой тайны, как мне их расставить на гуслях, чтоб играли так же волшебно, как у тебя". Отвечал купец Садко Морскому Царю: "Так нужно струны устанавливать: возьми сначала струну, которая дает самый высокий звук и установи ее на левый край гуслей. Потом из оставшихся струн снова возьми струну, которая дает самый высокий звук, и установи ее уже на правый край гуслей. Затем из оставшихся струн опять возьми ту, которая дает самый высокий звук, и установи ее на левый край гуслей. А потом из оставшихся струн возьми струну с самым высоким звуком, и установи ее на правый край гуслей. И так делай, пока все струны не установишь. Понял?". "Больно мудрено!" – сказал Морской Царь. – "Сам–то я не справлюсь, придется поручить кому-нибудь".

Напишите программу, которая поможет Морскому Царю установить струны на гуслях. Входными данными программы являются числа, которые отражают высоты звучания струн.

Входные данные

В первой строке файла входных данных записано целое число N – число струн, $1<=N<=100$. Далее следует N строк, в каждой из которых записано одно целое положительное число, не превышающее 100, – высота звучания струны с соответствующим номером. Чем больше это число, тем более высокий звук дает струна.

Выходные данные

В файл выходных данных требуется поместить числа высоты звучания струн, отражающие нужный порядок установки струн на гуслях. Числа следует отделять одним пробелом.

Примеры входных и выходных данных

INPUT.TXT	OUTPUT.TXT
7 10 8 4 12 6 1 3	12 8 4 1 3 6 10

Указания к решению

Для решения требуется аккуратно запрограммировать алгоритм, изложенный в тексте задачи. Заведем два массива – для хранения входных и выходных данных. Заметим, что заполнение выходного массива происходит подобно маятнику: на первом шаге заполняется первый элемент, на втором – последний, на третьем – второй элемент, на четвертом шаге – предпоследний элемент и т.д. Для заполнения берется максимальный элемент входного массива среди тех элементов, которые еще не были помещены в выходной массив. Нахождение максимального элемента оформим в виде подпрограммы. После того, как максимальный элемент помещен в массив выходных данных, в массиве входных данных присвоим ему значение –1, чтобы исключить его из дальнейшей обработки. Фрагмент решения:

```
const
        MaxN = 100;
type
        TArray = array [1..MaxN] of Integer;
function FindMax(A: TArray; Finish: Integer): Integer;
var
        M, Num, i: Integer;
begin
        Num := 1;
```

```
        M := A[1];
        for i := 1 to Finish do
                if A[i] > M then begin
                        Num := i;
                        M := A[i];
                end;
        FindMax := Num;
end;
var
        i, j, k, N: Integer;
        A1, A2: TArray;
begin
        …
        for i := 1 to N do begin
                if i mod 2 = 0 then
                        j := N – i div 2 + 1
                else
                        j := (i+1) div 2;
                k := FindMax(A1, N);
                A2[j] := A1[k];
                A1[k] := –1;
        end;
        …
end.
```

Эталонные тесты

№ п/п	INPUT.TXT	OUTPUT.TXT
1	7 10 8 4 12 6 1 3	12 8 4 1 3 6 10
2	1 31	31
3	14 2 5 9 1 3 6 8 0 11 4 12 98 67 45	98 45 11 8 5 3 1 0 2 4 6 9 12 67

Задача "Рация и чемодан"

Штирлиц поручил радистке Кэт подобрать чемодан для перевозки рации. Чемодан и рация имеют форму параллелепипеда. Рацию можно класть в чемодан только вертикально или горизонтально (рацию нельзя ставить под углом к дну чемодана). Боковые стенки рации должны быть параллельны боковым стенкам чемодана. Рация НЕ может поместиться в чемодане, если при этом какой–либо из ее размеров больше или равен соот-

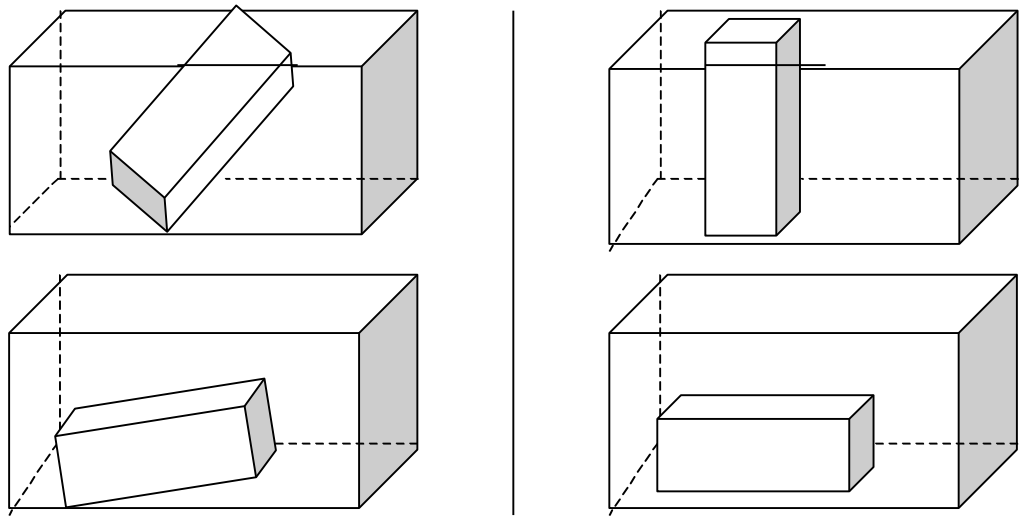
ветствующему размеру чемодана. Ниже на рисунке приведены примеры неправильного и правильного размещения рации в чемодане.

Напишите программу, которая подскажет радистке Кэт, поместится ли рация в чемодане.

Входные данные

В первой строке файла входных данных записано три числа – размеры рации в сантиметрах, а во второй – размеры чемодана в сантиметрах. Все числа целые положительные и не превосходят 100.

НЕПРАВИЛЬНО **ПРАВИЛЬНО**



Выходные данные

В файл выходных данных требуется записать строку ПОМЕЩАЕТСЯ, если рация помещается в чемодане, или строку НЕ ПОМЕЩАЕТСЯ в противном случае.

Примеры входных и выходных данных

INPUT.TXT	OUTPUT.TXT
10 15 5 20 40 30	ПОМЕЩАЕТСЯ
10 15 5 5 5 5	НЕ ПОМЕЩАЕТСЯ
10 15 5 15 10 5	НЕ ПОМЕЩАЕТСЯ
5 10 40 40 50 10	ПОМЕЩАЕТСЯ

Указания к решению

Для решения данной задачи требуется рассмотреть шесть различных случаев взаимного расположения рации и чемодана: три случая, когда чемодан неподвижен, а рация поворачивается на 90^0, и три случая, когда рация неподвижна, а на 90^0 поворачивается чемодан. Фрагмент решения:

```
var
        R1, R2, R3, S1, S2, S3: Integer;
begin
        …
        if ((R1 < S1) and (R2 < S2) and (R3 < S3)) or
           ((R3 < S3) and (R2 < S1) and (R1 < S2)) or
           ((R1 < S1) and (R2 < S3) and (R3 < S3)) or
```

```
        ((R2 < S2) and (R1 < S3) and (R3 < S1)) or
        ((R2 < S1) and (R3 < S2) and (R1 < S3)) or
        ((R1 < S2) and (R3 < S1) and (R2 < S3)) then
            WriteLn(F, 'ПОМЕЩАЕТСЯ')
    else
            WriteLn(F, 'НЕ ПОМЕЩАЕТСЯ');
    …
end.
```

Эталонные тесты

№ п/п	INPUT.TXT	OUTPUT.TXT
1	10 15 5 20 40 30	ПОМЕЩАЕТСЯ
2	10 15 5 5 5 5	НЕ ПОМЕЩАЕТСЯ
3	10 15 5 15 10 5	НЕ ПОМЕЩАЕТСЯ
4	5 10 40 40 50 10	ПОМЕЩАЕТСЯ
5	3 4 5 6 4 5	ПОМЕЩАЕТСЯ
6	30 40 50 50 40 30	НЕ ПОМЕЩАЕТСЯ
7	4 5 6 6 4 2	НЕ ПОМЕЩАЕТСЯ
8	14 10 1 14 11 15	ПОМЕЩАЕТСЯ

Задача "Кока-кола"

Представьте себе, что у Вас есть R рублей. Одна бутылка кока-колы стоит B рублей, одну пустую бутылку кока-колы Вы можете сдать в приемный пункт и получить E рублей, а емкость бутылки – C литров.

На все имеющиеся у Вас деньги Вы покупаете кока-колу, выпиваете ее (сами или с друзьями) и сдаете все пустые бутылки в приемный пункт. К полученным деньгам Вы добавляете те деньги, которые у Вас остались от первой покупки, и снова на все деньги покупаете кока-колу. Затем Вы ее снова выпиваете, сдаете все пустые бутылки в приемный пункт и так далее, пока у Вас не останется денег на покупку даже одной бутылки кока-колы.

Напишите программу, которая по заданным числам R, B, E, C определяет, сколько литров кока-колы можно выпить, действуя вышеуказанным способом.

Входные данные

В файле входных данных записано четыре целых числа (каждое – в отдельной строке):

R – начальное количество рублей, $0 <= R <= 32000$;

B – цена одной полной бутылки кока-колы в рублях, $1 <= B <= 32000$;

E – цена одной пустой бутылки кока-колы в рублях, $1 <= E <= 32000$, $E < B$ (иначе Вы будете пить кока-колу вечно!);

C – емкость одной бутылки Кока–колы в литрах, $1 <= C <= 10$.

Выходные данные

В файл выходных данных требуется записать одно целое число – количество литров кока-колы, которое можно выпить, действуя вышеуказанным способом.

Примеры входных и выходных данных

INPUT.TXT	OUTPUT.TXT
3 2 1 1	2
13 3 1 3	18
1 2 1 1	0

Указания к решению

Фрагмент решения:

```
var
        R, { было денег вначале }
        B, { цена одной бутылки }
        E, { цена пустой бутылки }
        C, { емкость бутылки }
        q, { столько бутылок купили на очередном шаге }
        Qty, { всего купили бутылок }
        L: { искомое число литров }
                LongInt;
begin
        …
        Qty := 0;
        while (R div B > 0) do begin
                q := R div B;
                R := q*E + R mod B;
                Qty := Qty + q;
        end;
        L := Qty*C;
        …
end.
```

Эталонные тесты

№ п/п	INPUT.TXT	OUTPUT.TXT
1	3 2 1 1	2
2	13 3 1 3	18
3	1 2 1 1	0
4	100 5 2 1	32
5	1250 43 20 3	159

№ п/п	INPUT.TXT	OUTPUT.TXT
6	32000 2 1 2	63998

Задача "MMM"

Для заданного натурального числа *число Мавроди* вычисляется следующим образом: определяется сумма цифр этого числа, затем сумма цифр полученного числа и так далее, пока не получится однозначное число. Например, для числа 7914 число Мавроди равно 3 (7+9+1+4=21, 2+1=3).

Для набора последовательных натуральных чисел *число MMM* вычисляется следующим образом: определяется число Мавроди для каждого числа из этого набора, затем все полученные числа Мавроди складываются, и вычисляется число Мавроди от полученной суммы. Например, для чисел 7914, 7915, 7916, 7917 число MMM равно 9 (числа Мавроди равны 3, 4, 5, 6, их сумма 3+4+5+6=18, число Мавроди от суммы 1+8=9).

Откуда такие названия? Однажды С. Мавроди, президент небезызвестной компании "MMM", сделал следующее. Он взял все денежные купюры, которые его компания собрала с доверчивых граждан, и решил подсчитать число MMM для номеров этих купюр. Хотя денег компания "MMM" с граждан собрала много (миллион купюр), Мавроди занимался этими подсчетами вручную. Поэтому компания "MMM" вскоре разорилась.

Напишите программу, которая спасет компанию "MMM" от разорения и **за 1 секунду** найдет число MMM. Входными данными программы являются номер первой и номер последней денежной купюры.

Входные данные

В файле входных данных записано два целых числа (каждое – в отдельной строке):

N1 – номер первой купюры, 1<=***N1***<=1 000 000;

N2 – номер последней купюры, 1<=***N2***<=1 000 000, ***N2***>***N1***.

Выходные данные

В файл выходных данных требуется записать одно целое число – число MMM.

Примеры входных и выходных данных

INPUT.TXT	OUTPUT.TXT
7914 7917	9
1 20	3
1 999999	9

Указания к решению

"Лобовое" решение (для каждого числа из заданного диапазона найти сумму цифр, если она больше 9, затем снова найти сумму цифр и т.д.) не проходит по времени на тесте, в котором задан диапазон номеров купюр от 1 до 1 000 000.

Вспомним признак делимости числа на 9: число делится на 9 одновременно с суммой цифр. А как связана с числом сумма его цифр, если число на 9 не делится? Рассмотрим произвольное число $a_n10^n+a_{n-1}10^{n-1}+\ldots+a_110+a_0$. Сумма его цифр равна $a_n+a_{n-1}+\ldots+a_1+a_0$. Найдем разность между числом и суммой его цифр, она равна $a_n(10^n-1)+a_{n-1}(10^{n-1}-1)+\ldots+a_1(10-1)$. Очевидно, что эта разность делится на 9. Следовательно, остатки от деления на 9 некоторого числа и суммы его цифр равны. В задаче требуется вычислять сумму цифр числа, пока не получится однозначное число. Значит, по

окончании этого процесса мы получим остаток от деления числа на 9, если число не делится на 9, и 9, если число делится на 9. Таким образом, если число делится нацело на 9, то число Мавроди от него равно 9, иначе оно равно остатку от деления исходного числа на 9.

Далее заметим, что числа Мавроди образуют периодическую последовательность вида 1, 2, 3, 4, 5, 6, 7, 8, 9, 1, 2, 3, 4, 5, 6, 7, 8, 9, … Количество повторений периода 1, …, 9 в этой последовательности равно *(N2–N1+1) div 9*. Количество оставшихся чисел, не составляющих период, равно *(N2–N1+1) mod 9*.

Период указанной последовательности представляет собой арифметическую прогрессию длины 9 и разностью 1 с первым членом, равным 1. С помощью формулы суммы арифметической прогрессии $S_n=(m_1+m_n)n/2$, где m_1 – первый член прогрессии, m_n – последний член прогрессии, находим, что сумма чисел в периоде равна 45.

Сумму чисел Мавроди, не составляющих период, найдем с помощью модификации формулы суммы арифметической прогрессии: так как $m_k=m_1+d(k–1)$, где d – разность прогрессии, то $S_n=(2m_1+d(n–1))n/2$. В данном случае первым членом прогрессии является *N1*, разность равна 1, а число членов в прогрессии равно *(N2–N1+1) mod 9*.

Таким образом, можно выписать следующее решение данной задачи:

```
function Mavrodi(N: LongInt): Integer;
begin
        if N mod 9 = 0 then Mavrodi := 9 else Mavrodi := N mod 9;
end;
var
        N, S1, S2, MMM: LongInt;
begin
        …
        N := N2–N1+1;
        S1 := (N div 9) * 45;
        S2 := ((2*N1 + N mod 9 – 1) * (N mod 9)) div 2;
        MMM := Mavrodi(S1+S2);
        …
end.
```

Эталонные тесты

№ п/п	INPUT.TXT	OUTPUT.TXT
1	7914 7917	9
2	1 20	3
3	1 999999	9
4	1 1000000	1
5	1 999998	9

Задачи олимпиады 2002 г.

Задача "Дроби"

Петя Бейсиков опять получил двойку по математике, так как неправильно решил примеры на арифметические действия с дробями.

Напишите программу, которая поможет Пете находить сумму, разность, произведение и частное дробей.

Входные данные

В файле входных данных записаны четыре целых числа из диапазона 0..1000 – числитель и знаменатель первой и второй дробей соответственно. Знаменатели не равны нулю. Далее записан знак арифметической операции:

+ сложение,
– вычитание,
***** умножение,
: деление.

Выходные данные

В файл выходных данных нужно записать решаемый пример и его результат в следующем виде:

`<1-я дробь><знак><2-я дробь>=<ответ примера>`

При записи дробей числитель и знаменатель отделяются знаком **/**. Ответ примера должен быть записан в виде несократимой дроби. При этом ненулевая целая часть ответа от дробной части должна отделяться запятой.

Примеры входных и выходных данных

INPUT.TXT	OUTPUT.TXT
1 3 1 3 +	1/3+1/3=2/3
1 3 1 3 :	1/3:1/3=1
1 3 2 3 -	1/3-2/3=-1/3
1 3 3 3 +	1/3+3/3=1,1/3

Указания к решению

Для решения данной задачи нам потребуется подпрограмма поиска наименьшего общего делителя (НОД) двух чисел – для нахождения общего знаменателя при действиях с дробями. Воспользуемся алгоритмом Евклида:

```
function NOD(m, n: LongInt): LongInt;
begin
        while not ((m=0) or (n=0)) do begin
                if m>=n then
                        m:=m–n;
                else
                        n := n – m;
        end;
        if m=0 then
                NOD:=n;
        else
                NOD:=m;
end;
```

Обратите внимание, что мы используем тип данных LongInt, чтобы в дальнейшем избежать ошибок выхода за границы диапазона типа Integer (например, при вычислении произведения дробей $^{998}/_{999}$ и $^{996}/_{997}$).

Далее, для сокращения дроби нам потребуется подпрограмма поиска наибольшего общего кратного (НОК) двух чисел, которое выражается через НОД:

```
function NOK(m, n: LongInt): LongInt;
begin
        NOD:=(m*n) div NOD(m,n);
end;
```

Подпрограмма нахождения суммы двух дробей:

```
procedure Plus(c1, z1, c2, z2: LongInt; var c, z: LongInt);
begin
        c1:=c1 div NOK(c1,z1);
```

```
        z1:=z1 div NOK(c1,z1);
        c2:=c2 div NOK(c2,z2);
        z2:=z2 div NOK(c2,z2);
        z:= NOD(z1,z2);
        c:=c1*(z div z1) + c2*(z div z2);
end;
```

Подпрограмма Mines нахождения разности двух дробей отличается от подпрограммы Plus только последним оператором: c:=c1*(z div z1) – c2*(z div z2);.

Подпрограмма нахождения произведения двух дробей:

```
procedure Mul(c1, z1, c2, z2: LongInt; var c, z: LongInt);
begin
        c1:=c1 div NOK(c1,z1);
        z1:=z1 div NOK(c1,z1);
        c2:=c2 div NOK(c2,z2);
        z2:=z2 div NOK(c2,z2);
        c1:=c1 div NOK (c1,z2);
        z2:=z2 div NOK(c1,z2);
        c2:=c2 div NOK (c2,z1);
        z1:=z1 div NOK(c2,z1);
        c:=c1*c2;
        z:=z1*z2;
end;
```

Частное двух дробей найдем как произведение первой дроби на дробь, обратную ко второй дроби:

```
procedure Divide(c1, z1, c2, z2: LongInt; var c, z: LongInt);
begin
        Mul(c1,z1,z2,c2,c,z);
end;
```

При выводе результата нужно рассмотреть отдельно случаи, когда числитель дроби равен 0, знаменатель дроби равен 1, числитель и знаменатель равны, а также выделить целую часть дроби, если ее числитель больше знаменателя.

Эталонные тесты

№ п/п	INPUT.TXT	OUTPUT.TXT
1	1 3 1 3 +	1/3+1/3=2/3
2	1 3 1 3 :	1/3:1/3=1
3	1 3 2 3 -	1/3-2/3=-1/3
4	1 3 3 3 +	1/3+3/3=1,1/3
5	2 7 5 9 +	2/7+5/9=53/63
6	7 12 5 8 -	7/12-5/8=-1/24
7	0 1 7 8 :	0/1:7/8=0
8	1000 1 1000 1 *	1000/1*1000/1=1000000
9	2 2 3 3 +	2/2+3/3=2
10	8 12 5 8 :	8/12:5/8=1,1/15
11	4 4 10 10 *	4/4*10/10=1
12	4 8 3 6 +	4/8+3/6=1

Задача "Покупка"

Экономный покупатель всегда покупает товар в оптовом магазине: когда товар покупаешь оптом, его цена уменьшается. Однако, чтобы уменьшить цену товара при оптовой покупке, экономный покупатель иногда должен купить товара несколько больше, чем ему требовалось. Но экономный покупатель этого не боится: он запаслив и бережлив, и излишек товара у него не пропадет.

Предположим, экономному покупателю нужно купить T тюбиков зубной пасты. Если покупать только один тюбик пасты, то его цена t рублей. Тюбики пакуются в пакеты по P тюбиков в пакете. Если покупать один пакет тюбиков, то цена одного тюбика в паке-

те будет ***p*** рублей, причем ***p***<***t***. Пакеты с зубной пастой пакуются в коробки по ***K*** пакетов в коробке. Если покупать одну коробку тюбиков, то цена одного тюбика в коробке будет ***k*** рублей, причем ***k***<***p***.

Например, пусть цена отдельного тюбика пасты 10,5 руб., цена тюбика в пакете 8,5 руб., а в одном пакете 12 тюбиков. Тогда экономный покупатель вместо 10 тюбиков пасты (стоимость покупки 10*10,5=105 руб.) должен купить 1 пакет тюбиков (стоимость покупки 8,5*12=102 руб.), из которого 2 тюбика (12–10=2) у него останутся про запас.

Напишите программу, которая подскажет экономному покупателю, сколько коробок, пакетов и тюбиков пасты ему следует купить, чтобы общая стоимость покупки была наименьшей, а также подсчитает эту стоимость и количество тюбиков, которые останутся у покупателя про запас.

Входные данные

В первой строке файла входных данных записано число ***t*** (цена отдельного тюбика пасты). Во второй строке – числа ***p*** (цена одного тюбика в пакете) и ***P*** (количество тюбиков в пакете). В третьей строке – числа ***k*** (цена одного тюбика в коробке) и ***K*** (количество пакетов в коробке). В четвертой строке записано число ***T*** – количество тюбиков, которое нужно купить экономному покупателю.

Числа ***k***, ***p*** и ***t*** – вещественные положительные, причем ***k***<***p***<***t***<=1 000. Числа ***K***, ***P*** и ***T*** – целые положительные, причем ***K***, ***P***, ***T***<=50 000.

Выходные данные

В файл выходных данных требуется записать результаты вычислений в следующем формате:

```
<количество коробок>
<количество пакетов>
<количество отдельных тюбиков>
<общее количество тюбиков>
<стоимость покупки >
<количество тюбиков про запас>
```

Примеры входных и выходных данных

INPUT.TXT	OUTPUT.TXT
10.5 8.5 12 8 12 1	0 0 1 1 10.50 0
10.5 8.5 12 8 12 10	0 1 0 12 102.00 2
10.5 8.5 12 8 12 9	0 0 9 9 94.50 0

Указания к решению

Алгоритм решения следующий. Сначала нужно найти оптимальную покупку (без излишков). Затем определить, возможно ли ее удешевить одним из двух способов: взять лишний пакет и не брать отдельных тюбиков либо взять лишнюю коробку и не брать ни пакетов, ни отдельных тюбиков. Фрагмент решения:

```
var
        T, { количество тюбиков, которое нужно купить покупателю }
        P, { количество тюбиков в пакете }
        K, { количество пакетов в коробке }
        BOX, { количество купленных коробок }
        PKG, { количество купленных пакетов }
        PCS, { количество купленных отдельных тюбиков }
        TOT_QTY, { общее количество купленных отдельных тюбиков }
        RESERVE { количество тюбиков, купленных про запас }
                : LongInt;
        tt, { цена отдельного тюбика }
        pp, { цена одного тюбика в пакете }
        kk, { цена одного тюбика в коробке }
        TOT_SUM { общая сумма покупки }
                : Real;
begin
        …
        BOX:=T div (K*P);
        PKG:=(T mod (K*P)) div P;
        PCS:=(T mod (K*P)) mod P;
        if PCS*tt > P*pp then begin
                PCS:=0;
                PKG:=PKG+1;
        end;
        if PKG*P*pp+PCS*tt>K*P*kk then begin
                PKG:=0;
                PCS:=0;
                BOX:=BOX+1;
        end;
        TOT_QTY:=BOX*K*P+PKG*P+PCS;
        RESERVE:=TOT_QTY–T;
        TOT_SUM:=BOX*K*P*kk+PKG*P*pp+PCS*tt;
        …
end.
```

Эталонные тесты

№ п/п	INPUT.TXT	OUTPUT.TXT
1	10.5 8.5 12 8 12 1	0 0 1 1 10.50 0
2	10.5 8.5 12 8 12 10	0 1 0 12 102.00 2

№ п/п	INPUT.TXT	OUTPUT.TXT
3	10.5 8.5 12 8 12 9	0 0 9 9 94.50 0
4	10.5 8.5 12 7.91 12 131	0 11 0 132 1122.00 1
5	10.5 8.5 12 7.91 12 134	1 0 0 144 1139.04 10
6	10.5 8.5 12 8.4 12 50000	347 2 8 50000 395534.88 0
7	15 8.5 12 8 12 50000	347 3 0 50004 400050.00 4

Задача "Планирование"

Когда Петя Бейсиков принес домой "двойки" по математике, русскому, литературе и географии, его папа сказал:

— Хватит! Теперь я буду контролировать, как ты делаешь уроки. Немедленно составь план выполнения домашних заданий по этим предметам на следующую неделю. Для каждого предмета отметь начало и завершение выполнения домашних заданий. Используй следующие обозначения:

Предмет	Начало выполнения домашнего задания	Завершение выполнения домашнего задания
Математика	**(**	**)**
Русский язык	**{**	**}**
Литература	**[**	**]**
География	**<**	**>**

Твой план должен подчиняться следующим правилам:

1. Каждое начатое домашнее задание нужно завершить.
2. Нельзя завершать домашнее задание, которое не начато.
3. Если начато более одного домашнего задания, то завершать нужно сначала то, которое было начато последним.

Напишите программу, которая поможет папе проверить составленный Петей план выполнения домашних заданий.

Входные данные

В файле входных данных записана одна строка – непустой план выполнения домашних заданий. План состоит из условных обозначений, указанных выше, а заканчивается точкой, обрабатывать которую не нужно. Длина строки не превышает 200.

Выходные данные

Если план выполнения домашних заданий удовлетворяет всем правилам, то в файл выходных данных требуется БОЛЬШИМИ буквами записать строку `YES`, иначе записать строку `NO`.

Примеры входных и выходных данных

INPUT.TXT	OUTPUT.TXT
`().`	`YES`
`(.`	`NO`
`).`	`NO`
`({<[]>}).`	`YES`
`({<[>]}).`	`NO`
`((({{[[<>]]}}))){[]<()>}.`	`YES`
`((({{[[<>]]}}))){[]<>()}.`	`NO`

Указания к решению

Алгоритм решения основан на использовании структуры данных стек. *Стек* – это массив, в котором просмотр, добавление и удаление элементов осуществляется только в один конец, называемый *вершиной* стека. Для работы со стеком можно использовать только следующие операции: "поместить элемент (в вершину стека)", "удалить элемент (находящийся на вершине стека)", "посмотреть элемент (находящийся на вершине стека)", "проверить стек на пустоту".

Идея алгоритма состоит в следующем. Мы последовательно просматриваем входную строку. Если встречается открывающая скобка, мы помещаем ее в стек. Если встречается закрывающая скобка, то мы удаляем элемент из стека. При этом, если удаленная из стека открывающая скобка не соответствует только что прочитанной закрывающей скобке (например, `[` и `}`), то входная строка ошибочна. Входная строка будет верной, если по окончании указанных действий стек окажется пустым.

Далее приведем полное решение:

```
Program Planning;
var
        F: Text;
        Stack: array [1..200] of Char;
        StackPtr: Integer;
        Ch: Char;
        Error: Boolean;
procedure Stack_Init;
begin
        StackPtr:=0;
end;
function Stack_Pop: Char;
begin
        Stack_Pop := Stack[StackPtr];
        StackPtr := StackPtr – 1;
end;
function Stack_Top: Char;
begin
        Stack_Top := Stack[StackPtr];
end;
function Stack_IsEmpty: Boolean;
```

```
begin
        Stack_IsEmpty := StackPtr = 0;
end;
function Stack_IsFull: Boolean;
begin
        Stack_IsFull := StackPtr = MaxStackLen;
end;
procedure Stack_Push(Ch: Char);
begin
        StackPtr := StackPtr + 1;
        S[StackPtr] := Ch;
end;

begin
        …
        Ch := '+';
        Error := False;
        while (Ch<>'.') and (not Error) do begin
                Read(F, Ch);
                if Ch in ['<', '(', '{', '[' ] then
                        Stack_Push(Ch)
                else
                        if Ch in ['>', ')', '}', ']' ] then
                                if Stack_IsEmpty then begin
                                        Error := True;
                                        break;
                                end
                        else
                        begin
                                case Stack_Pop of
                                        '<': if Ch<>'>' then Error := True;
                                        '(': if Ch<>')' then Error := True;
                                        '{': if Ch<>'}' then Error := True;
                                        '[': if Ch<>']' then Error := True;
                                end;
                                if Error then break;
                        end;
        end;
        Error := Error or (not Stack_IsEmpty);
        if Error then WriteLn(F, 'NO') else WriteLn(F, 'YES');
end.
```

Эталонные тесты

№ п/п	INPUT.TXT	OUTPUT.TXT
1	`().`	YES
2	`(.`	NO
3	`).`	NO
4	`({<[]>}).`	YES
5	`({<[>]}).`	NO
6	`((({{[[<>]]}}))){[]<()>}.`	YES
7	`((({{[[<>]]}}))){[]<>()}.`	NO
8	`(({{{[[<<>>]]}}})){}[]<>.`	YES
9	`{}}{><][(.`	NO

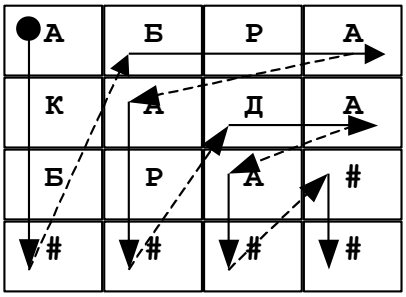
Задача "Шифровка"

Агент ФБР Малли получил задание зашифровать сообщение для агента Скалдера. Шифрование выполняется в следующем порядке (пусть шифруется сообщение, состоящее из одного слова АБРАКАДАБРА):

1. Подбирается наименьший квадратный листок клетчатой бумаги, где можно записать сообщение по одному символу в каждой клетке. В нашем случае длина сообщения равна 11, и листок имеет размер 4*4.
2. В каждой клетке листка записывается один символ сообщения в порядке слева направо и сверху вниз. Если некоторые клетки листка остаются пустыми, то они заполняются символом **#**. В нашем случае листок будет заполнен так:

А	Б	Р	А
К	А	Д	А
Б	Р	А	#
#	#	#	#

3. Затем символы сообщения записываются с листка по одному в строку в порядке, который указан на рисунке стрелками:



То есть сначала записывается 1 столбец, затем остаток 1 строки, далее остаток 2–го столбца, после этого остаток 2 строки и т.д., пока не будут выписаны все символы. В нашем случае зашифрованное сообщение имеет вид АКБ#БРААР#ДАА###

Напишите программу, которая поможет агенту Малли зашифровать заданное сообщение.

Входные данные

В файле входных данных записано непустое сообщение, которое нужно зашифровать. Оно состоит только из букв и цифр, его длина не превышает 200. Сообщение может быть записано на нескольких строках, в этом случае символы конца строки и перевода строки шифровать не нужно. Признаком окончания сообщения является точка, которую шифровать не нужно.

Выходные данные

В файл выходных данных следует записать в строку зашифрованное сообщение.

Примеры входных и выходных данных

INPUT.TXT	OUTPUT.TXT
АБРАКАДАБРА.	АКБ#БРААР#ДАА###
АБРА КА ДАБРА .	АКБ#БРААР#ДАА###

Указания к решению

Для решения данной задачи требуется аккуратно запрограммировать алгоритм, изложенный в тексте задачи. Заведем два массива: одномерный – для хранения исходного сообщения, и двумерный – для хранения зашифрованного сообщения в виде матрицы.

```
var
    Msg: array [0..200] of Char;
```

```
Cipher: array [0..20, 0..20] of Char;
```

Пусть N – длина исходного сообщения, тогда размерности матрицы для хранения зашифрованного сообщения найдем с помощью следующей функции:

```
function CipherDim(N: Byte): Byte;
var
    D: Integer;
begin
    D:=Trunc(Sqrt(N+1));
    if N+1 > Sqr(D) then D:=D+1;
    CipherDim:=D;
end;
```

Фрагмент реализации шифрования сообщения (N – длина исходного сообщения, D – размерность матрицы для хранения зашифрованного сообщения):

```
for i:=0 to D–1 do
    for j:=0 to D–1 do
        if i*D+j <= N then
            Cipher[i,j] := Msg[i*D+j]
        else
            Cipher[i,j] := '#';
```

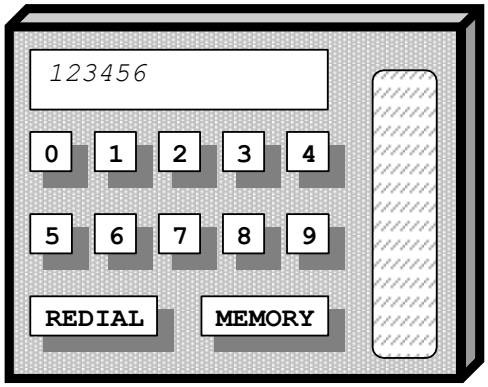
Фрагмент вывода зашифрованного сообщения в выходной файл:

```
for i:=0 to D–2 do begin
    for j:=i to D–1 do
        Write(F, Cipher[j,i]);
    for j:=i+1 to R–1 do
        Write(F, Cipher[i,j]);
end;
Write(F, Cipher[D–1, D–1]);
```

Эталонные тесты

№ п/п	INPUT.TXT	OUTPUT.TXT
1	АБРАКАДАБРА.	АКБ#БРААР#ДАА###
2	АБРА КА ДАБРА .	АКБ#БРААР#ДАА###
3	А.	А
4	ABCDE.	AD#BCE###
5	1234567890 .	16161616161616#23456789012345727 2727272727#89012345678903838383 3838#456789012345949494949494#012 34567890505050505#6789012345161 61616##23456789072727##8901234 5383838##456789094949##012345505 0##67890161##234572##8903##45### #

Задача "Телефон"



Современные телефоны, помимо обычных кнопок–цифр, имеют также две специальные кнопки: **REDIAL** и **MEMORY**.

Кнопка **REDIAL** выполняет повтор ранее набранного номера. Если нажать кнопку **REDIAL** и удерживать ее некоторое время, то телефон повторит набор последнего набранного номера. Если нажать кнопку **REDIAL**, а затем цифру **2**, то телефон выполнит набор предыдущего набранного номера. Телефон сохраняет список последних набранных номеров, и аналогичным образом можно набрать любой из них. Например, чтобы набрать 20-й номер из этого списка, нужно нажать кнопку **REDIAL**, а затем цифры **2** и **0**. При наборе номера он становится первым в списке последних набранных номеров. Длина этого списка ограничена, и при переполнении из него удаляется номер, попавший туда первым. Один и тот же номер не может входить в список последних набранных номеров более одного раза.

Телефон имеет память, в которой постоянно хранится список номеров для ускоренного набора. Кнопка **MEMORY** выполняет ускоренный набор номера из этого списка. Например, чтобы набрать 1-й номер, нужно нажать кнопку **MEMORY**, а затем цифру **1**. Один и тот же номер не может входить в список номеров для ускоренного набора более одного раза.

Секретарь фирмы "Покупки по телефону" ежедневно получает список телефонных номеров клиентов фирмы, которым надо позвонить. Однако номера в списке часто повторяются, и набирать их полностью (то есть нажимать все кнопки номера) очень утомительно.

Фирма наняла Вас в качестве программиста, чтобы Вы разработали программу в помощь секретарю. Входными данными программы являются список телефонных номеров постоянных клиентов фирмы, размер списка последних набранных номеров и список телефонных номеров, по которым звонит секретарь. Программа должна выдать такую последовательность нажатий кнопок телефона, чтобы выполнить все звонки в заданном порядке, но количество нажатых кнопок было бы наименьшим из возможных.

Примечание. Если один и тот же номер можно набрать двумя способами, используя одинаковое количество нажатых кнопок (с помощью кнопки **REDIAL** и с помощью кнопки **MEMORY**), то следует выбрать способ набора с кнопкой **MEMORY**. Например, если один и тот же номер можно набрать как **REDIAL 20** и **MEMORY 11**, то следует выбрать способ **MEMORY 11**.

Входные данные

В первой строке файла входных данных записано целое число M – размер списка номеров, которые постоянно хранятся в памяти телефона для ускоренного набора с помощью кнопки **MEMORY**, $1 <= M <= 100$. Далее идут M строк с соответствующими телефонными номерами.

Затем в отдельной строке записано целое число R– размер списка последних набранных номеров, $1 <= R <= M$.

После этого в отдельной строке записано целое число L – размер списка телефонных номеров, по которым должен позвонить секретарь, $1 <= L <= 1000$. Далее идут L строк с соответствующими телефонными номерами.

Все телефонные номера представляют собой 6-значные целые положительные числа.

Выходные данные

В файл выходных данных требуется записать последовательность нажатий кнопок телефона, чтобы выполнить все звонки из входного списка в заданном порядке, причем количество нажатых кнопок было бы наименьшим из возможных. Названия кнопок **`REDIAL`** и **`MEMORY`** записываются БОЛЬШИМИ буквами. Если за названием одной из этих кнопок следует номер, то он отделяется одним пробелом.

Примеры входных и выходных данных

INPUT.TXT	OUTPUT.TXT
1 111111 1 1 222222	222222
3 111111 222222 333333 3 5 444444 555555 111111 222222 111111	444444 555555 MEMORY 1 MEMORY 2 MEMORY 1
2 111111 333333 2 9 111111 111111 222222 222222 111111 222222 333333 222222 222222	MEMORY 1 REDIAL 222222 REDIAL MEMORY 1 REDIAL 2 MEMORY 2 REDIAL 2 REDIAL

Указания к решению

Задача является технически сложной: алгоритм нахождения последовательности нажатий кнопок телефона однозначно описан в условии задачи, однако он содержит достаточно большое количество условий и оговорок и потому непрост для реализации.

Далее в указаниях мы будем использовать следующие обозначения:

- PHONE – заданный список телефонных номеров, PHONE (number) – порядковый номер элемента number в заданном списке телефонных номеров;
- REDIAL – список ранее набранных телефонных номеров, REDIAL (number) – порядковый номер элемента number в списке ранее набранных телефонных номеров;
- MEMORY – список телефонных номеров для ускоренного набора, MEMORY (number) – порядковый номер элемента number в списке телефонных номеров для ускоренного набора;
- BEST – наилучшая последовательность нажатий кнопок телефона.

Нахождение BEST предполагает последовательный просмотр PHONE. При этом возможны следующие случаи:

1. Текущий элемент PHONE отсутствует в REDIAL и MEMORY.
   В этом случае текущий элемент PHONE заносится в BEST.
2. Текущий элемент PHONE отсутствует в REDIAL, но присутствует в MEMORY.
   В этом случае в BEST заносится пара { кнопка_MEMORY, MEMORY (текущий элемент PHONE) }.
3. Текущий элемент PHONE отсутствует в MEMORY, но присутствует в REDIAL.
   В этом случае в BEST заносится пара { кнопка_REDIAL, REDIAL (текущий элемент PHONE) }.
4. Текущий элемент PHONE присутствует в REDIAL и MEMORY.
   Если REDIAL (текущий элемент PHONE)=1, то в BEST заносится пара { кнопка_REDIAL, REDIAL (текущий элемент PHONE) }, то есть текущий элемент PHONE был набран на предыдущем шаге, и для его повтора достаточно нажать лишь одну кнопку Redial.
   В противном случае определяются два числа: r – количество цифр в REDIAL (текущий элемент PHONE) и m – количество цифр в MEMORY (текущий элемент PHONE).
   Если r<m, то в BEST заносится пара { кнопка_REDIAL, REDIAL (текущий элемент PHONE) }, иначе в BEST заносится пара {кнопка_MEMORY, MEMORY (текущий элемент PHONE) }.

Кроме того, после обработки очередного элемента списка PHONE необходимо перестроить список REDIAL следующим образом. Если первый элемент списка REDIAL не совпадает с очередным элементом списка PHONE, то очередной элементом списка PHONE помещается на первое место в REDIAL, а остальные элементы REDIAL сдвигаются (при этом последний элемент REDIAL, возможно, теряется).

Эталонные тесты

Первые эталонные тесты совпадают с примерами входных и выходных данных, приведенными в условии задачи. Для генерации остальных тестов использовалась следующая программа:

```
Program TelephoneTestGenerator;
const
        MaxM = 100;
        MaxR = MaxM;
        MaxL = 1000;
var
        F: Text;
        i, M, R, L: LongInt;
begin
        Randomize;
        Assign(F, 'input.txt');
        Rewrite(F);
        M := 1+Random(MaxM–1);
        WriteLn(F, M);
        for i:=0 to M–1 do
                WriteLn(F, (1+Random(9))*100000+(1+Random(9))*10000);
        R := 1+Random(M–1);
        WriteLn(F, R);
        L := 1+Random(MaxL–1);
        WriteLn(F, L);
        for i:=0 to L–1 do
                WriteLn(F, (1+Random(9))*100000+(1+Random(9))*10000);
        Close(F);
end.
```

5. Положение об олимпиаде

1. Общие положения

Олимпиада по программированию для школьников 5–8 классов проводится ежегодно Управлением по делам образования, Челябинским государственным университетом и лицеем № 11 г. Челябинска.

Участие школ города в олимпиаде организуется по командному принципу. Каждый район города имеет право выставить для участия в соревнованиях команду одной школы.

1.1. Цели олимпиады

Основными целями олимпиады являются:

- стимулирование интереса школьников к углубленному изучению программирования и информатики;
- выявление школьников, обладающих выраженными способностями в области программирования и информатики;
- развитие интеллектуального и творческого потенциала школьников.

1.2. Порядок организации и проведения олимпиады

1. Для участия в олимпиаде от района выдвигается одна школа, команда которой будет представлять район на городской олимпиаде.
2. Для участия в городском этапе каждая школа–участник должна до 15 марта подать в городской оргкомитет олимпиады заявку по установленной форме (см. ниже).

1.3. Организационное обеспечение олимпиады

1. Каждому участнику во время проведения олимпиады выделяется отдельный персональный компьютер и комплект заданий.
2. Во время выполнения заданий участник может использовать принесенные с собой письменные принадлежности и чистые листы бумаги. Использование печатных, электронных и любых других источников информации запрещается.

2. Участники олимпиады

1. Олимпиада проводится для учащихся 5–8 классов общеобразовательных школ, в том числе экспериментальных площадок, лицеев.
2. Каждая команда–участник должна состоять из трех человек, обучающихся в одной школе. Возрастной состав команды не регламентируется.
3. Руководителем команды является преподаватель соответствующей школы, который несет полную ответственность за жизнь и здоровье детей во время проведения олимпиады.

3. Руководство и методическое обеспечение олимпиады

1. Общее руководство олимпиадой осуществляется городским оргкомитетом олимпиады, который состоит из специалистов Управления по делам образования, Челябинского государственного университета и лицея № 11.
2. Городской оргкомитет олимпиады:
   - согласует форму и порядок проведения олимпиады на всех ее этапах;
   - оказывает методическую помощь районным оргкомитетам олимпиады;
   - формирует состав жюри, которое разрабатывает задания заключительного этапа

олимпиады;

- анализирует и обобщает итоги олимпиады.

3. Персональный состав городского оргкомитета:
   - В.Д. Батухтин, ректор ЧелГУ, профессор, д-р физ.-мат. наук – сопредседатель;
   - А.Г. Гостев, директор МОУ лицей № 11, профессор, д-р пед. наук – сопредседатель;
   - В.Н. Кеспиков, начальник Управления по делам образования г. Челябинска, канд. пед. наук – сопредседатель;
   - М.И. Солодкова, зам. начальника Управления по делам образования г. Челябинска по научно-методическому обеспечению системы образования
   - О.С. Виноградова, начальник отдела Управления по делам образования г. Челябинска;
   - Л.Б. Соколинский, зав. Управлением информатизации образования, зав. кафедрой математического обеспечения ЭВМ ЧелГУ, канд. физ.-мат. наук, доцент;
   - К.Р. Овчинникова, доцент кафедры математического обеспечения ЭВМ ЧелГУ, канд. пед. наук;
   - М.Л. Цымблер, доцент кафедры математического обеспечения ЭВМ ЧелГУ, канд. физ.-мат. наук;
   - И.В. Курбатова, зав. Центром информатизации образования, лицей № 11;
   - Т.Г. Яковлева, зав. кафедрой информатики и экономики, лицей № 11.
4. Координаты городского оргкомитета:

Адреса	ул. Тимирязева, дом 6	ул. Бр. Кашириных, дом 129
Телефоны	66–08–07	42–04–09
E–mail	cio@cio11.schel.ac.ru	mzym@csu.ru, of@csu.ru

4. Техническое обеспечение олимпиады

1. Олимпиада проводится на базе лицея № 11 и Челябинского государственного университета.
2. Специальное программное обеспечение, необходимое для проведения олимпиады, на безвозмездной основе предоставляет Челябинский государственный университет.

5. Финансовое обеспечение олимпиады

Финансирование олимпиады осуществляют Управление по делам образования, Челябинский государственный университет и лицей № 11 с возможным привлечением спонсорских средств.

6. Подведение итогов олимпиады

1. Итоги олимпиады подводятся в личном и командном зачетах. Подведение итогов олимпиады в личном зачете осуществляется автоматизированной программной системой по правилам, утверждаемым городским оргкомитетом олимпиады. В командном зачете итоги олимпиады подводятся на основе результатов, показанных участниками команды в личном зачете. Результаты олимпиады объявляются в день проведения соревнований.
2. В личном зачете итоги олимпиады подводятся раздельно для учащихся 5–6 и 7–8 классов. В каждой подгруппе дипломами Управления по делам образования г. Челябинска награждаются учащиеся, занявшие первые три места в своей подгруппе.

Кроме того, учащийся, занявший 1–3 место в подгруппе, обеспечивает своей школе в городской олимпиаде следующего года одно дополнительное место участника.

3. В командном зачете итог выступления школы подсчитывается как среднее арифметическое общего количества задач, решенных участниками команды. При равенстве данного показателя вторым показателем для выявления победителя в командном зачете является количество занятых участниками команды первых мест, третьим показателем – количество вторых мест и т.д. В командном зачете дипломами оргкомитета соответствующих степеней награждаются первые три команды-победительницы.

7. Задания олимпиады

Задания для олимпиады готовятся городским оргкомитетом и являются едиными для всех участников.

Каждое задание включает в себя 4–6 задач, предполагающих составление программы на любом из предложенных языков программирования (перечень допустимых языков программирования см. ниже).

Заявка на участие в олимпиаде по программированию среди школьников 5–8 классов г. Челябинска

1. Школа _____
2. Директор (Ф.И.О. полностью) _____
3. Адрес школы _____
4. Телефон _____
5. Факс _____
6. E–mail _____
7. Информация о команде (на каждую команду заполняется отдельный раздел)
   - 7.1. Информация о руководителе команды
     - 7.1.1. Ф.И.О. полностью _____
     - 7.1.2. Должность _____
     - 7.1.3. Телефон рабочий _____
     - 7.1.4. Телефон домашний _____
     - 7.1.5. E-mail _____
   - 7.2. Информация о каждом участнике команды
     - 7.2.1. Ф.И.О. полностью _____
     - 7.2.2. Класс _____
     - 7.2.3. Год рождения _____
     - 7.2.4. Домашний адрес, телефон _____
     - 7.2.5. E–mail _____

Дата заполнения _____

Директор _____

Руководитель команды _____

ПЕРЕЧЕНЬ допустимых языков программирования

При решении задач олимпиады по программированию допускается использование любого из следующих языков программирования: Pascal, C, C++, BASIC.

Содержание

ГОРОДСКАЯ ОЛИМПИАДА ШКОЛЬНИКОВ 5–8 КЛАССОВ ПО ПРОГРАММИРОВАНИЮ 2001–2002 ГГ.

Редактор В.Ф. Репецкая

Компьютерная верстка М.Л. Цымблер

Подписано в печать 04.04.2003. Формат 60×84 $^{1}/_{16}$.
Бумага офсетная. Печать офсетная.
Усл. печ. л. 2,1. Уч.-изд. л. 3,1.
Тираж 130 экз. Заказ 78. Цена договорная

Челябинский государственный университет
454021 Челябинск, ул. Братьев Кашириных, 129

Полиграфический участок Издательского центра ЧелГУ
454021 Челябинск, ул. Молодогвардейцев, 57б