Задача №1 - римские цифры

Важно! В этой задаче не нужно писать программу.
Составьте набор входных и выходных данных, позволяющих протестировать работу программы, которая записывает числа римскими цифрами. Программу писать в этом задании не нужно.

Вам дана эталонная (правильная) программа **Roman.exe**, которая выводит число римскими цифрами в ответ на число, введённое арабскими цифрами.
https://ru.wikipedia.org/wiki/Римские цифры

Формат входных и выходных данных программы Roman.exe

Программа считывает целое число и выводит его, записанное римскими цифрами.
либо ERROR, если было введено не целое число, либо оно вне диапазона от 1 до 3999

Примеры входных и выходных данных

Ввод	**Вывод**
17	XVII
abc	ERROR

Указания

Изучите как работает данная вам **правильная программа** Roman.exe. Запустите её и посмотрите какие данные она выведет в ответ.
Предоставьте папку с набором файлов с именами **input<N>.txt** и **output<N>.txt**, где <N> – некоторое натуральное число. Файлы **input<N>.txt** содержат входные данные, а **output<N>.txt** ожидаемые выходные данные.
Например в файле *output3.txt* запишите данные, которые правильная программа должна вывести, если ей подать на вход данные из файла *input3.txt*.
Созданные вами файлы будут использованы для проверки работы правильных и неправильных версий программы Roman.exe.
Правильная программа – та, которая выдаёт результат в соответствии с требованиями..
Неправильная программа – та, которая содержит выдаёт результат, отличный от ожидаемого.
Набор подготовленных вами файлов должен быть таким, чтобы:

- **все правильные программы** для каждого `input<N>.txt` вывели результат, совпадающий с `output<N>.txt`;
- **каждая из неправильных программ** для данных из хотя бы одного из входных input<N>.txt вывела результат, не совпадающий с output<N>.txt.

Рекомендации

Разместите ваши файлы со входными и выходными данными в папке tests, давая им имена:
input1.txt и output1.txt, input2.txt и output2.txt, и так далее, пока не решите, что написали достаточно тестовых данных.
В файлах input*.txt разместите входные данные, а в соответствующих им output*.txt - данные, которые правильно работающая программа должна вывести при этих входных данных.

Чтобы проверить, действительно ли ваша программа ведёт себя верно на тестовых данных, запустите в консоли файл test.bat с параметром, задающим номер теста.
Например:
`test.bat 1`
Проверит, что программа при передаче ей данных из tests\input1.txt выведет текст, равный содержимому файла tests\output1.txt.
Если выведется OK, значит всё хорошо. В противном случае будет выведено сообщение об ошибке.

Запуск test.bat без указания номера теста, равнозначен запуску теста №1.

Задача №2 - Угадай слово

Вступление

Ученик школы программирования Иннокентий договорился с преподавателями, что играть он будет только в те компьютерные игры, которые сам написал. Одна из любимых игр Иннокентия – “Угадай слово”. Помогите Иннокентию написать программу, позволяющую играть в эту игру.

Требования к программе

На вход в первой строке подаётся **Секретное слово** из 5 букв. Гарантируется, что все буквы заглавные латинские (от A до Z).
Затем программа запрашивает не более 6 слов по одному в строке. После ввода слова она выводит это же слово, добавляя после каждого символа:

- Символ "+", если буква стоит в той же позиции, что и в Секретном слове.
- Символ "*", если буква содержится в Секретном слове, но в другой позиции
- Символ "-", если этой буквы в Секретном слове нет

Если игрок ввёл Секретное слово, то нужно сразу вывести сообщение “**SUCCESS**”. В противном случае, по истечении 6 попыток, вывести сообщение “**YOU LOSE**”.
Если игрок ввёл Секретное Слово длиной, не равной 5 буквам, нужно вывести сообщение “**ERROR**” и завершить программу.
Если игрок, угадывая, ввёл пустую строку или слово, длина которого не равна 5 буквам, необходимо вывести сообщение “**INVALID WORD LENGTH”** и продолжить считывание слов. Попытка при этом не тратится.

Важно: необходимо соблюдать требуемый формат вывода. Программы, выводящие данные в неправильном формате будут не пройдут автоматические проверки и будут признаны неправильными.

Примеры входных и выходных данных.
Данные программа должна выводить именно так, как показано

Ввод	**Вывод**	**Пояснение**
HOUSE CLASS HOUSE	C-L-A-S+S* SUCCESS	Сперва пользователь ввёл секретное слово. Одна из S на своём месте, вторая - не на своём, C, L, A - отсутствуют Повезло, так повезло
CLASS MOUSE LASER CLEAR CLASS	M-O-U-S+E- L*A*S*E-R- C+L+E-A*R- SUCCESS	Сперва пользователь ввёл секретное слово. S на своём месте. Остальных букв нет. L, A, S есть, но в других позициях. C и L на своём месте, A не на своём. Загаданное слово найдено.
CLASS WORD LASER IMAGE CLEAR SOLVE BREAD MINUS	INVALID WORD LENGTH L*A*S*E-R- I-M-A+G-E- C-L+E-A*R- S*O-L*V-E- B-R-E-A*D- M-I-N-U-S+ YOU LOSE	Пользователь ввёл секретное слово. Слово не подходит, так как его длина не равна пяти буквам Несмотря на некорректное введенное слово, попытка не была потрачена. Игрок вводит ещё шесть слов, перед тем как проиграть.
WORD	ERROR	Пользователь ввёл секретное слово некорректной длины. Программа выводит сообщение об ошибке и прекращает работу.

Задача №3 - сложение чисел в произвольной системе счисления

Напишите программу, реализующую суммирование 2-х положительных чисел в любой системе счисления (до 36-ричной). Системы счисления, начиная с 11-ричной, используют **заглавные буквы латинского алфавита** для обозначения цифр с 10 до 35. Например, в 16-ричной системе счисления используются цифры от 0 до 9 и буквы от A до F.

Входные данные для программы
Программе на вход подаются 3 строки:

- В первой строке задается основание системы счисления – целое число от 2 до 36. Гарантируется, что в качестве системы счисления введено число в диапазоне от 2 до 36.
- Во второй и третьей строках задаются слагаемые – числа, записанные в указанной системе счисления. Числа могут иметь неодинаковое количество разрядов.

Требования к выходным данным программы
Программа должна вывести сумму введённых чисел в указанной системе счисления.

Если введённые числа содержали больше 254 разрядов, были пустыми строками либо содержали недопустимые в заданной системе счисления символы, программа должна вывести сообщение об ошибке "**ERROR!**" (заглавными буквами с восклицательным знаком).
Важно: необходимо соблюдать требуемый формат вывода. Программы, выводящие данные в неправильном формате будут не пройдут автоматические проверки и будут признаны неправильными.

Примеры возможных входных и выходных данных

Ввод	**Вывод**	**Пояснение**

10
9
9
	18	`  9`
		`+ 9`
		`---`
		` 18`

2
11001
0011
	11100	` 11001`
		`+ 0011`
		`------`
		` 11100`

16
F
F2
	101	В 16-ричной системе счисления используются цифры от 0 до F:
		`   F`
		`+ F2`
		`----`
		` 101`

2 **3** **2**	**ERROR!**	Числа в двоичной системе счисления могут содержать только цифры 0 и 1

Подсказки

Используйте строковое представление чисел, работая с отдельными символами строк как с разрядами чисел.

Используйте принцип сложения чисел “в столбик”. В любой системе счисления сложение в столбик происходит по одним и тем же правилам.

Строки можно склеивать при помощи операции “**+**”:

```
Str1 := 'Hello';
Str2 := 'World';
Str3 := Str1 + ' ' + Str2;
WRITELN(Str3);                  { Напечает Hello World }
```

К отдельным символам строки можно обращаться по индексу (порядковый номер символа, начиная с 1):

```
Str := 'Hello';
WRITELN(Str[2]); { напечатает символ e }
```

Длину строки можно узнать при помощи функции `LENGTH`:

```
Str := 'Hello';
WRITELN(LENGTH(Str)); { напечатает 5 }
```