Базовые задачи по языку РЕФАЛ

В следующих задачах требуется читать входные данные из файла input.txt, а результат записывать в файл output.txt.

1. Линейное уравнение (Взята!)

Задано арифметическое выражение с помощью операций '+','−', круглых скобок, целых чисел и переменной X. Задача: выяснить, при каких значениях X значение выражения будет равно 0.

Техническое требование

Выражение задаётся строкой с клавиатуры. Длина строки может быть не более 80 символов. Сумма любых чисел в выражении не превышает 2 000 002 001.

Ответ выдавать в виде обыкновенной дроби со знаком, где числитель и знаменатель разделены символом '/'. Числитель и знаменатель не должны иметь общих делителей, кроме единицы.

Пример

```
Выражение: -(-(X+(-50))-(X-6))
X=+28/1
```

2. Сокращение выражения (Взята!)

Задано арифметическое выражение с помощью операций '+', '−', '∗' и круглых скобок. Запишите выражение, эквивалентное заданному, но без скобок и с минимальным количеством операций. Выражения эквивалентны, если при любых значениях одинаковых переменных в них они дают одинаковые значения. Заглавные и строчные буквы не различаются.

Пример

```
Исходное выражение: ((123-X)*(2+3))-(X-(-Y))
Результат: 615-6*X-Y
```

3. Превращение программы в почтовое сообщение

Напишите программу, печатающую свой собственный исходный текст с добавлением в начало каждой строки символа '>'. Например, ваша программа была написана на каком-то языке программирования. Затем её оттранслировали. Результатом работы оттранслированной программы должен быть исходный текст вашей программы с добавленными в начало каждой строки символами '>'.

Техническое требование

Программа должна работать без внешних модулей и файлов. Результат работы записывается в файл *.TXT в текущей директории, где * — имя Вашей программы. Например, программа INPUT.PAS должна выдавать файл INPUT.TXT

4. Форматер программ (Взята!)

Напишите программу, на вход которой подается текст со сбалансированными фигурными скобками. На выходе — этот же текст, структурированный отступами слева с учетом вложенности Этих скобок.

5. Одинаковые программы (Взята!)

Можно сократить текст программы, если убрать символы, не влияющие на её работу (лишние пробелы, скобки и т.п.). Заметим, что добавлять ничего нельзя. Две программы назовём *одинаковыми*, если после удаления описанных выше элементов получится один и тот же текст. По двум заданным программам определить одинаковы ли они.

6. Логический калькулятор (Взята!)

Запрограммируйте вычисление значения логического выражения классической логики высказываний. 1 — обозначает истину, 0 — ложь.

Пример

Выражение: $1\&(1 => 0) \vee (0 => 1)$
0

7. Сжатие строки (Взята!)

Один из простых способов сжатия информации заключается в поиске одинаковых частей, идущих подряд. Вместо нескольких повторений можно записать их количество и всего одну часть.

Задача. По заданной строке выдать её максимальное сжатие. Длина получившейся строки вычисляется вместе с числами, обозначающими количества повторений. Длина числа — количество цифр в десятичной записи.

Техническое требование

Задана строка из латинских больших и маленьких букв. Длина входной строки не более 90 символов. Если вариантов сжатия несколько, то выдать любой.

Пример

```
Строка: abababABabAbbcccccccccccc
Сжатие: 4ab1ABabAbb11c
```

8. Максимальная последовательность символов (Взята!)

Напишите программу, которая запрашивает имя файла и печатает длину самой большой последовательности из одного и того же символа.

Пример

В тексте задачи самая длинная последовательность "мм" и длина её 2.

9. Предложение (Взята!)

Есть ли в файле заданное предложение? Предложения сравниваются с точностью до разделяющих пробелов, переходов на новую строку и табуляций. Знаки препинания — все символы, отличные от букв (русских или латинских).

Напоминание: знаки препинания могут выполнять роль разделителей. Переносов в словах не будет.

Пример

Предложение:

```
Знаки препинания-все символы отличные от
    букв    (  русских или латинских   )     .
```

```
Есть в тексте задания.
```

10. Дифференцирование (Взята!)

Задана функция, зависящая от x. Вычислите её производную.

Пример

```
F(x) =  (1-x)/g(x)
F'(x)= ((x-1)*g'(x)-g(x))/g^2(x)
```

11. Удаление символов (Взята!)

Можно ли из одной строки получить другую, убрав некоторые символы?

Пример

программа
гамма
`Результат: ДА`

12. Кратчайший делитель файла

Файл — конечная последовательность (цепочка) символов. Конкатенация файлов — сцепление этих последовательностей в одну. Умножение файла на число n — n-кратная конкатенация файла с самим собой. Кратное файла — результат его умножения на натуральное число. Делитель файла — файл, по отношению к которому исходный файл является кратным. Кратчайший делитель — наиболее короткий делитель.

Задача — найти кратчайший делитель исходного файла, если он есть.

Входные данные Файл *text.txt* — исходный файл.

Выходные данные Файл *div.txt* — выходной файл. Если делителей нет, сообщение об ошибке «делителей нет» выдается на монитор.

Ограничения Входной файл может содержать только цифры и латинские буквы (строчные и прописные). Время работы — не более 10 секунд. Объем обрабатываемого (за это время) файла — предмет конкурса.

Пример

Если *text.txt* — это цепочка
aabaabbaabaabbaabaabb

то *div.txt* — это
`aabaabb`

13. Палиндром перестановками (Взята!)

За какое минимальное число перестановок символов можно превратить заданную строку в палиндром? Перестановка — обмен местами двух разных символов.

Техническое требование

Программа должна читать символы из текстового файла *string.txt* и печатать минимальное число перестановок. Во входном файле все символы записана подряд. Символов не больше 10000. Если строку в палиндром превратить невозможно, то напечатать -1. Время будет ограничено.

Пример

Во входном файле: `ппоот`

```
Перестановок: 2
```

14. Уравнение f(X) = A (Взята!)

Решите уравнение вида $f(X) = A$, где A — строка из маленьких латинских букв, X — переменная строка, f — функция на строках, построенная с помощью операций $*$ и $+$ над строками. Операция $+$ обозначает конкатенацию строк, $A + B + C = (A + B) + C$. Например, $adbc + 123 = abcd123$. Результатом операции $A * B$ будет строка, где после каждого символа A записана B; если A или B — пустые строки, то результат будет также пустой строкой, $A*B*C = (A*B)*C$. Например, $ab * cd = acdbcd$. Операции выполняются в порядке их следования.

Техническое требование

Программа должна запросить уравнение и напечатать одно из решений. Строка может содержать разделительные пробелы. Проверка на синтаксическую правильность не требуется.

Пример

```
Уравнение: X*ab + (c * X) = aabcabcac
X=ac
```

15. Сравнение с образцом (Взята!)

Образец имеет вид:

```
образец ::= <PT>
  <PT> ::= <A><PT> | <A>
  <A>  ::= '['<PT>']' | <ST>
```

```
<ST> ::= <символы>
```

Часть образца, заключенная в квадратные скобки показывает, что эту часть можно исключать. Таким образом, образец описывает множество строк. Образец всегда правильный. Например:

образец: [a[b]]c

описывает строки: c, ac, abc

Слово подходит под образец, если оно совпадает с одним из слов, описываемых образцом. Задача написать программу, которая по заданным образцу и слову печатает «НЕТ», если слово подходит под образец, и «ДА», если не подходит.

Пример

```
образец: [a[b]]c
слово: ab
ответ: ДА
```

16. Палиндром из трёх частей (Взята!)

Задана строка символов. Можно ли разделить строку на три части, так чтобы из них можно было составить палиндром?

Палиндром — строка, которая пишется одинаково справа налево и слева направо.

Написать программу, которая запрашивает строку и печатает части, на которые предлагается делить (каждую часть с новой строки) и сам палиндром. Возможно, что некоторые части будут пустой строкой. Если вариантов ответа несколько, то выдать любой. Использовать нужно все три части. Если получить палиндром не удаётся, выдать "Невозможно".

Пример

```
Строка: АЛАШШ
Ш
Ш
АЛА
ШАЛАШ
```

17. Генетика на чужой планете

На планете Красивый Фингал, открытой петербургскими космонавтами, генетический код определяется тридцатью одним кодоном. Два из

них определяют структуру гена и обозначаются '(' и ')'. Три из них определяют преобразования генетического кода и обозначаются I, K и S. Все остальные, как обычно, порождают организм. Они обозначаются малыми латинскими буквами. Вначале геном некоторое время преобразуется, а затем начинает расти организм, поэтому, например, у свинофингала может родиться собакофингал, а то и дубофингал. Правила преобразований следующие.

```
((X)Y) в (XY), (1)
(XY) в ((X)Y), (2)
(IA) в A,
(KAB) в A,
(SABC) в ((AC)(BC)).
```

Здесь A, B, C — либо кодоны, либо последовательность кодонов, сбалансированная по скобкам и заключенная в скобки, X и Y — произвольные сбалансированные по скобкам непустые последовательности кодонов. Если к некоторому фрагменту было применено преобразование (1), то к нему уже не может быть применено преобразование (2), и наоборот.

Генетический код называется стабильным, если он не может преобразовываться, кроме как по правилам (1) и (2).

Если генетический код превысит в длину 250 символов, развитие зародыша прекращается. Точно так же оно прекращается, если код преобразуется более 100 шагов.

Задание Даны два входных файла. *Species.txt* содержит пары строк, описывающих геномы некоторых стабильных видов на планете. Первая, третья и так далее строки — имена видов, вторая строка — геном первого вида, четвертая — второго и так далее. Количество видов не более 8, длина каждой строки до 80 символов. Конец ввода — пустая строка. Биологи, как им и положено, пишут названия видов на варварской латыни, так что в строках могут быть лишь латинские буквы.
Пример файла:

```
Swintus Fingalae
(swintuS)
Canis Fingali
((((ca)n)I)s)
Chimera Baldina
(SSSSS)
```

Embrion.txt состоит из одной строки с генетическим кодом эмбриона. Для эмбриона нужно вывести историю преобразования и определить,

какому виду он принадлежит.
Пример файла:

```
(SKKs(SKIw)(IIIi)(KnK)tu)S
```

В файл *output.txt* нужно вывести результат, какому виду принадлежит эмбрион, либо квалификацию генетического кода. Квалификация может иметь следующие формы:

- новый вид, когда преобразование заканчивается, но получившийся геном принадлежит неописанному стабильному виду;
- опасный, когда преобразование приводит к слишком длинному коду или может продолжаться более 100 шагов;
- фатальный, когда преобразование приводит к печальному исходу с гарантией, по вашему мнению.

Ваш вывод должен быть обоснован предъявленным преобразованием. Файл заканчивается пустой строкой.

Пример

Выходной файл для данного случая.

```
Swintus Fingalae
((SKKs(SKIw)(IIIi)(KnK)tu)S)
(((SKKs)(SKIw)(IIIi)(KnK)tu)S)
((((Ks)(Ks))(SKIw)(IIIi)(KnK)tu)S)
(((Ks(Ks))(SKIw)(IIIi)(KnK)tu)S)
((s(SKIw)(IIIi)(KnK)tu)S)
(s(SKIw)(IIIi)(KnK)tuS)
(s(SKIw)(IIIi)ntuS)
(s(SKIw)((II)Ii)ntuS)
(s(SKIw)(IIi)ntuS)
(s(SKIw)((II)i)ntuS)
(s(SKIw)(Ii)ntuS)
(s(SKIw)intuS)
(s((Kw)(Iw))intuS)
(s(Kw(Iw))intuS)
(swintuS)
```

Внимание! Биологи, изучавшие планету, не очень подкованы в математике и программировании. Поэтому некоторые виды могут быть описаны неправильно, соответствующие геномы нестабильны; для таких геномов нужно вывести в файл *output.txt* ошибку, написав строчки вида:

```
Геном Chimera Baldina неверен
(SSSSS)
((SSSS)S)
(((SS)(SS))S)
***
```

18. Интерпритатор НАМ (Взята!)

Реализовать интерпритатор нормальных алгоритмов маркова (НАМ). Во входном файле правилам вида $\alpha \to \beta$ соответствуют три строки:
α
$>$
β
а правилам вида $\alpha \to .\ \beta$ соответствуют три строки:
α
.
β

В выходной файл нужно записать проткл работы НАМ на первой строке входного файла строке.

Пример

Input.txt:
1010201
01
>
10
2
.
Error
Output.txt:
01012
10012
10102
11002
1100Error01

```
.oooO Oooo.
(   ) (   )
 \ (   ) /
  \_) (_/
```

Рис. 2: Следы

19. Следы (Взята!)

Будем изображать следы ног человека с помощью обычных символов на клавиатуре следующим образом (рис. 2).

Задача: заполнить следами N человек, идущих рядом и проходящих путь в S полушагов. $S < 12$; $N < 12$.

Пример

Для $N = 3$ и $S = 5$.

```
.oooO     .oooO     .oooO
(   )     (   )     (   )
 \ ( Oooo. \ ( Oooo. \ ( Oooo.
  \_)(   )  \_)(   )  \_)(   )
.oooO ) / .oooO ) / .oooO ) /
(   )(_/  (   )(_/  (   )(_/
 \ ( Oooo. \ ( Oooo. \ ( Oooo.
  \_)(   )  \_)(   )  \_)(   )
.oooO ) / .oooO ) / .oooO ) /
(   )(_/  (   )(_/  (   )(_/
 \ (       \ (       \ (
  \_)       \_)       \_)
```

20. Наибольшая общая подстрока (Взята!)

Заданы две строки. Найти третью, которая является подстрокой двух заданных и имеет максимально большую длину.

Пример

Строка 1: *ababab*
Строка 2: *bababa*
ответ: ababa

21. Тождественность выражений (Взята!)

Заданы записи двух арифметических выражений с целыми числами, операциями сложения, вычитания, умножения, деления нацело, круглыми скобками и переменными. Напишите программу, которая проверяет тождественность выражений, то есть совпадают ли значения выражений для всех возможных значений переменных.

22. Требуется налить (Взята!)

Даны n сосудов, имеющих объемы v_1, v_2, ..., v_n, где v_i — целые числа, обозначающие литры. Ответьте на вопрос, можно ли с помощью указанных сосудов отмерить ровно v литров жидкости?

23. Одна строка

Дан текстовый файл, состоящий из n строк ($1 \leqslant n \leqslant 50000$), длина каждой строки не превосходит 255 символов. В файле есть ровно одна строка, которая встречается ровно один раз, все остальные строки повторяются. Найдите эту уникальную строку.

24. Абракадабра (Взята!)

Последовательность строк $s_0, s_1, s_2, s_3 \ldots$ строится следующим образом: s_0— пустая строка, а каждая последующая строка получается удваиванием предыдущей строки и приписыванием к ней слева очередной буквы латинского алфавита (начиная с буквы a). Таким образом,
$s_0 =$''; $s_1 =$'a'; $s_2 =$'baa'; $s_3 =$'$cbaabaa$'; ...
Даны натуральные числа n, m ($1 \leqslant n \leqslant 26; 1 \leqslant m \leqslant 2^n - 1$). Определите символ, который стоит в m-ой позиции в строке s_n.

25. Бусы (Взята!)

Сколько разных бус можно собрать из заданного набора разноцветных бусинок? Бусы — это цепочка бусинок, надетых на нитку, которая связана в кольцо, поэтому у них нет начала и конца. Понятно, что как бусы ни поворачивай — это одни и те же бусы. Если поменять местами бусинки одного цвета, бусы от этого не изменятся.

Техническое требование

Задана строка латинских букв, больших и маленьких. Каждая буква обозначает цвет. Причём маленькая буква обозначает темный вариант, а большая — светлый (r — красный, R — светло-красный). Бусинок одного цвета столько, сколько раз встречается эта буква в строке. Длина строки не более 50 символов.

Пример

Бусинки: *tSst*
Разных бус: 2

26. Вычёркивание чисел (Взята!)

Вычеркните из числа

12345...99100101...200220032004

k-значное число (незначащие нули входят), чтобы оставшееся число было максимальным из возможных.

27. Конструкции фигурок (Взята!)

Задана строка не более, чем из 1000 символов. Одинаковыми символами в ней задаются конструкции фигурок. *Конструкции фигурок* — это только расстояния, на которых расположены друг от друга символы, но не сами символы. Таким образом, в строке могут быть записаны одинаковые конструкции фигурок разными символами.

Техническое требование

Напишите программу, которая по заданной строке выводит число разных конструкций фигурок.

Для записи строки будут использованы только обычные текстовые символы. Входная строка находится в файле *string.txt*. Время работы — 5 секунд.

Пример

В *string.txt*:
abc␣abc␣abc
Ответ:
2

28. Квадратная спираль (Взята!)

Квадратная таблица размером 40000×40000 заполнена последовательными натуральными числами, начиная с 1, по спирали. Спираль начинается в левом верхнем углу с координатам (1, 1) и закручивается по часовой стрелке внутрь. Например, спираль 3×3 выглядит так, как показано на рисунке 3.

1	2	3
8	9	4
7	6	5

Рис. 3: Спираль 3×3

Техническое требование

Написать программу, которая по заданным координатам (строка, столбец) выводит число в клетке с этими координатами.

Пример

```
Координаты: 2   10
160005
```

29. Жулик на тестах

Лаборатория по обработке текстов разработала язык программирования СТРОКА. Этот язык предназначен для обработки строк. Для данного языка создан интерпретатор ***string***.

Среди сотрудников лаборатории решено провести турнир по владению этим языком. Для этого решено дать какую-то задачу, а решения проверять только по тестам. После того как задача и тесты были готовы, один из сотрудников сумел раздобыть тесты к задаче. Помогите сотруднику выступить в этом конкурсе хорошо.

Синтаксис языка СТРОКА Все команды записываются в виде: A r B, где A, B — строки или переменные, r — операция языка. Команда означает, что нужно вычислить результат операции r для операндов A и B, после чего записать этот результат в переменную Z.

Переменными языка являются только заглавные латинские буквы. Переменные A и Z имеют специальные значения. A — хранит входную цепочку. Значение Z будет выведено после окончания работы интерпретатора и считается результатом.

Константами языка являются любые строки, заключённые в кавычки. Строки в тестах будут текстовые, без управляющих символов и одинарных кавычек.

Команды этого языка записываются по одной в строке.

Имеются следующие операции:

$x + y$ — сцепление строк (конкатенация);

$x - y$ — удаление максимальной хвостовой части x совпадающей с некоторой головной частью y. Например, `'abcd'-'bcdb'` $\rightarrow$ `'a'`;

$x|y$ — результатом будет строка, полученная удалением из x всех символов, которых нет в y на том же расстоянии от начала. Например, `'abcd'|'accbef'` $\rightarrow$ `'ac'`;

$x\&y$ — результатом будет строка, полученная удалением из x всех символов, которые стоят в y на том же расстоянии от начала. Например, `'abcd'&'accbef'` $\rightarrow$ `'bd'`;

$x = y$ — замена символов строки x на последовательности символов y до исчерпания x. Например, `'abcdeadfrtg'='123'` $\rightarrow$ `'12312312312'`;

$X < Y$ — поместить значение переменной Y в X.

Задача Написать программу для построения программы на языке СТРОКА по известным тестам. Тесты будут находится в файле *test.txt*. Построенная программа записывается в файл *program.str*. Она должна проходить как можно больше тестов.

Формат записи тестов в файле *test.txt* В файле с тестами каждая нечетная строка — это входные данные, а четная — выходные.

Техническое требование

Тестов в файле будет не больше 10. Таким образом, количество строк — до 20. Размер файла с построенной программой не должен превышать 100 Мб. Каждая строка — до 200 символов. Количество строк не ограничивается. Константы будут текстовые, без управляющих символов. Вторая одинарная кавычка обозначает конец константы. Время работы программы не должно превышать 50 секунд на Р-200. Памяти, дисковой и оперативной, будет выделено не менее 1 Мб.

Пример

Тесты

```
abc
b
aaaaa
aaa
123456
2345
```

Программа

```
'1234' = ' '
Z < A
Z & '1-----'
Z & '----6'
Z & 'a----'
Z & '-a--'
Z & '-c'
```

30. Интерпретатор String (Взята!)

Написать интерпретатор языка СТРОКА (см. задачу).

31. Интерпретатор String-2 (Взята!)

Написать интерпретатор языка СТРОКА-2 (см. задачу).

32. СТРОКА-2

Лаборатория по обработке текстов разработала язык программирования СТРОКА-2. Этот язык предназначен для обработки строк. Он устроен так же, как язык СТРОКА (см. задачу). Для данного языка создан интерпретатор ***string***.

Синтаксис языка СТРОКА-2 Все команды записываются в виде: A r B, где A, B — строки или переменные, r — операция языка. Команда означает, что нужно вычислить результат операции r для операндов A и B, после чего записать этот результат в переменную Z.

Переменными языка являются только заглавные латинские буквы. Переменные A и Z имеют специальные значения. A — хранит входную цепочку. Значение Z будет выведено после окончания работы интерпретатора и считается результатом.

Константами языка являются любые строки. Константы заключаются в кавычки. Строки в тестах будут текстовые, без управляющих символов и одинарных кавычек.

Команды этого языка записываются по одной в строке.

В отличие от языка СТРОКА, в языке СТРОКА-2 имеются только одна операция:

$x < y$ — заменить первую слева подстроку y в z на x.

Задача Напишите программу для построения программы на языке СТРОКА-2 по известным тестам. Тесты будут находится в файле *test.txt*. Построенная программа записывается в файл *program.str*. Она должна проходить как можно больше тестов.

Формат записи тестов в файле *test.txt* В файле с тестами каждая нечетная строка — это входные данные, а четная — выходные.

Техническое требование

Тестов в файле будет не больше 10000, т.е. строк до 20000. Длины строк до 100 символов. Размер файла с построенной программой не должен превышать 100 Мб. Каждая строка до 200 символов. Количество строк не ограничивается. Константы будут текстовые, без управляющих символов. Вторая одинарная кавычка обозначает конец константы. Время работы программы не должно превышать 50 секунд на Р-200. Памяти дисковой и оперативной будет выделено не менее 1 Мб.

Пример

Тесты

```
abc
b
aaaaa
aaa
123456
2345
```

Программа

```
Z < A
'abc' < 'b'
'aaaaa' < 'aaa'
'123456' < '2345'
```

33. Частые числа (Взята!)

Строится последовательность чисел:

$$\begin{aligned} X_1 &= (N+1)\#(N+1), \\ &\dots \\ X_i &= (X_{i-1}+i)\#(X_{i-1}+i), \\ &\dots \\ X_N &= (X_{N-1}+N)\#(X_{N-1}+N). \end{aligned}$$

Выдайте число, которое встречается чаще других в этой последовательности. Если таких чисел несколько, то выдайте самое маленькое из них. Число $0 < N < 100\,000$ задаётся пользователем.

Операция # берёт от левого числа последние две цифры, от правого — первые две и соединяет их в одно число. Если цифр не хватает, то берут незначащие нули (слева). Например, $12345\#12345 = 4512$, $1\#2 = 0102$

Пример

```
N= 10

Ответ:  1111
```

34. Перемешанная строка (Взята!)

Под *перемешиванием строки* будем понимать следующий процесс:

1) берём заданную строку;

2) если символов нечётное количество, то добавляем в начало строки пробел, получаем исходную[1] строку;

3) делим строку пополам, получаем правую и левую строки;

4) новая строка получается сцеплением правой, исходной и левой строк (правая оказалась слева, левая — справа);

При повторении результат берётся в качестве исходной строки. Например, для строки «строчка» последовательность такова:

1) строчка

2) ␣строчка

3) ␣стр очка

4) очка␣строчка␣стр

Описанную последовательность выполняем 2004 раз. Определите в получившейся строке символ, который будет находиться на 2004 месте.

Техническое требование

Длина входной строки не превышает 100 символов. На вход подаются только непустые строки.

Пример

```
Строка: строчка
На 2004 месте находится символ: а
```

35. Подбор умножений (Взята!)

В строке A расставить '*' так, чтобы получилась строка B. '*' — умножение по Пупышеву. Когда * несколько, то вначале выполняется самое левое, а затем остальные.

[1] Обратите внимание, что в этой задаче «исходная строка» и «заданная строка» означают разные понятия.

Техническое требование

В файле записаны две строки, первая A, вторая B. Выдать нужно решение, либо «нет решений», если вариант расстановки * найти невозможно. Длины строк не более 5000.

Пример

A= *abcd*

B= *adcdbdcd*

Ответ: ab*c*d

36. Уравнение X*(A*X+B) = C (Взята!)

Для уравнения $X * (A * X + B) = C$, где A, B, C, X строки. Операция * имеет более высокий приоритет чем +. Скобки применяются так же как в обычной арифметике. Найти X.

Техническое требование

A, B, C записаны в файл. Выдать нужно выдать X либо «нет решений», если такой X найти невозможно. Длина A, B и C не более 10000.

Пример

A = *aaa*

B = *bb*

C = *aaabcaabcaabcbbbaabcaabcaabcbbcaabcaabcaabcbb*

X = abc