Всесибирская олимпиада по программированию. Заочный тур.

Решения задач необходимо направить по адресу: suncngu@mail.ru

В письме надо указать фамилию, имя, отчество, дату рождения, класс (или номер вашей учебной группы), название учебного учреждения, его полный адрес или прикрепить файл с анкетными данными.

К письму нужно приложить как исходный текст, так и откомпилированную программу. При проведении заочного тура Всесибирской олимпиады школьников по информатике допустимыми языками программирования являются C, C++, Pascal, Visual Basic. Возможно использование следующего программного обеспечения:

- Borland Delphi 7.0
- FreePascal 2.2.0
- Microsoft Visual C/C++ 2005
- Microsoft Visual Basic 2005
- Borland Pascal 7.0
- Borland C/C++ 3.1

Программы должны называться: фамилия, имя, знак подчеркивания, номер задачи.

Например, ИвановСаша_3.exe ; ИвановСаша_3.pas .

Работу можно сдавать частично и до окончательно срока. Промежуточные рейтинги будут публиковаться.
Окончательный срок сдачи работ 15 **января**.

Необходимо строго соблюдать формат входного и выходного файла, т.к. проверка будет проводиться автоматически. Ни в коем случае нельзя использовать процедуры и функции, ожидающие ввода с клавиатуры (readkey, getch), так как в этом случае программа будет ждать ввода бесконечно (и будет снята с тестирования после превышения лимита времени).
Существует простой способ чтения из файла и записи в файл:

Pascal:

```
assign(input,'input.txt');
reset(input);
assign(output,'output.txt');
rewrite(output);
{ теперь обычные функции read, readln, write, writeln будут работать с файлами, а не с клавиатурой / экраном }
{ в конце программы } close(output);
```

C:

```
freopen("input.txt", "r", stdin);
freopen("output.txt", "w", stdout);
/* теперь обычные функции printf, scanf, puts, fgetc и т.п. будут работать с файлами, а не с клавиатурой / экраном */
```

Для всех задач:

Имя входного файла:	`input.txt`
Имя выходного файла:	`output.txt`
Ограничение по памяти:	`64 Мб`
Максимальная оценка за задачу:	`100 баллов`

Задача 1. Площадь многоугольника. Геометрический алгоритм.

Ограничение по времени: `1 секунда на тест`

Во входном файле располагаются координаты точек, являющихся соседними вершинами многоугольника. Предполагается, что первая вершина соединена с последней вершиной. Требуется написать программу, вычисляющую площадь многоугольника и в выходной файл вывести удвоенную величину этой площади.

Входные данные

В первой строке входного файла располагается число m - количество вершин многоугольника ($2 < m \le 1000$).

В следующих m строках располагаются пары целых чисел X и Y – координаты вершин ($-1000 \le X,Y \le 1000$). Периметр многоугольника задается без самопересечений.

Выходные данные

В выходной файл требуется вывести единственное целое число – удвоенную площадь многоугольника.

Примеры

input.txt	*output.txt*
3 1 2 3 4 4 1	8

Методические указания

Разбить фигуру на множество трапеций. Площадь каждой трапеции можно вычислить по формуле $s=(Y_k+Y_{k+1})*(X_k-X_{k+1})/2$

Задача 2. Длинная арифметика.

Ограничение по времени: `1 секунда на тест`

Вычислите факториал числа N. N!=1*2*3*4*5*...*(n-1)*n.

Входные данные

Во входном файле располагается одно целое число N ($1 \le N \le 3000$).

Выходные данные

В выходной файл требуется вывести значение N! в виде строки цифр.

Примеры

input.txt	*output.txt*
5	120

Методические указания

Храните цифры числа в отдельных ячейках массива. Подробнее смотрите здесь.

Задача 3. Конечный автомат.

Ограничение по времени: `1 секунда на тест`

Уберите из заданного текста комментарии. В данной задаче комментарий - это текст, заключенный в фигурные скобки {}, а также текст, заключенный в скобки из двух символов (* *) или текст, следующий после пары символов // и до конца строки (не включая символы конца строки). Конец строки кодируется двумя последовательными символами #13 и #10. При решении задачи разрешается использовать только две ячейки памяти типа byte. Файл прочитывается один раз и в памяти не хранится. Внешние файлы не создаются. Комментарии не могут быть вложенными друг в друга. Указание: если скобка, открывающая комментарий, встретилась, то далее следует комментарий, пока не встретится соответствующая закрывающая скобка, конец строки или конец файла.

Входные данные

Во входном файле располагается текст. Конец текста определяйте по функции «конец файла». Конец строки – по паре символов конца строки. Размер исходного файла не превышает 1 Мбайт.

Выходные данные

В выходной файл требуется вывести текст без комментариев.

Примеры

input.txt	*output.txt*
Akjn {dfn} n Wf //eff Wfw (*{kuh**) wef	Akjn n Wf Wfw wef

Методические указания

Постройте диаграмму состояний и переходов. В одной ячейке храните текущий символ из файла, в другой номер состояния. Подробнее смотрите в Википедии «Конечный автомат».

Задача 4. Работа со стеком.

Ограничение по времени: 1 секунда на тест

Определите правильность скобочного выражения, состоящего из скобок двух видов ( ) и [ ]. У правильного скобочного выражения внутри каждой пары, состоящей из открывающей и закрывающей скобки одного вида, все прочие скобки должны быть закрыты.

Входные данные

Во входном файле располагается строчка символов, состоящая из скобок. Длина файла не превышает 1 Мбайт.

Выходные данные

В выходной файл требуется вывести 1, если скобочное выражение правильное, или 0, если наоборот.

Примеры

input.txt	*output.txt*
())))(()	0
([(()]))(	0
(([]))[]	1
(([[()]]))[[()]]	1
[(])	0

Методические указания

Если скобка открывающая, то записывайте скобку в стек, если закрывающая, то извлекайте скобку из стека и сравнивайте. Подробнее смотрите в Википедии «Стек».

Задача 5. Рекурсия.

Ограничение по времени: 1 секунда на тест

Вычислить значение арифметического выражения. В выражении используются арифметические операторы: сложение, вычитание и умножение. Если не стоят скобки, то у умножения приоритет выше, чем у сложения и вычитания. Разрешается знак минус, меняющий на обратное значение как целого выражения, так и отдельного числа. Арифметические операторы не могут стоять рядом (подряд) - обязательно отделяются скобками. Операнды (то над чем совершаются арифметические действия) – целые числа от 0 до 9.

Входные данные

Во входном файле располагается строчка символов. Длина не превышает 255 символов. Все арифметические выражения корректны. Значения выражений, как промежуточные, так и конечные, по модулю не превышают 2147483647.

Выходные данные

В выходной файл требуется вывести значение выражения.

Примеры

input.txt	*output.txt*
-(9-8)*3	-3
((1+2)*(3-4)+2)*(-1)	1

Методические указания

В данной задаче предлагается применить рекурсию для вычисления значения выражения, так как любая часть выражения это тоже выражение.

Например, если f – значение выражения в скобках, то f(‘2+3*4’) =f(‘2’)+f(‘3*4’) , а в свою очередь f(‘3*4’)=f(‘3’)*f(‘4’). Подробнее смотрите в Википедии «Рекурсия».

Задача 6. Полный перебор.

Ограничение по времени: `1 секунда на тест`

Определите количество вариантов решения ребуса. Наш ребус – это выражение в виде суммы двух чисел. Некоторые цифры чисел заменены на латинские буквы. Вам необходимо найти все варианты этих замен так, чтобы арифметическое выражение становилось верным. В данной задаче варианты решения ребуса a+b=3 такие, как 1+2=3 и 2+1=3, считаются разными. Первые цифры чисел не могут равняться нулю. Так, например, ребус abcd+befd=edfbg имеет три решения: 9382+3152=12534, 5782+7192=12974 и 9675+6185=15860.

Входные данные

Во входном файле располагается одна строчка символов. Длина строки не превышает 255 символов. Символ «+» разделяет слагаемые, символ «=» отделяет результат.

Выходные данные

В выходной файл требуется вывести количество вариантов решения ребуса.

Примеры

input.txt	*output.txt*
`1274+5430=6704`	`1`
`a1+2b=cd`	`15`
`abcd+befd=edfbg`	`3`

Методические указания

При полном переборе вариантов используйте отсечения. Так чтоб отсечения были эффективны, начинайте перебор с младших разрядов. Подробнее смотрите в Википедии «Полный перебор».

Задача 7. Обход графа в ширину.

Ограничение по времени: `1 секунда на тест`

Найдите кратчайший путь в лабиринте между двумя заданными клетками. Лабиринт задан на клеточном поле. На одни клетки можно наступать, на другие нет. Переходить можно на соседние свободные клетки, соприкасающиеся сторонами.

Входные данные

Во входном файле в первой строчке располагаются шесть чисел. Первое число Xmax – это размер лабиринта по горизонтали ($3 \le Xmax \le 255$). Второе число Ymax – размер лабиринта по вертикали ($3 \le Ymax \le 100000$). Третье и четвертое число – координаты начальной клетки. Пятое и шестое число – координаты конечной клетки. Первое число в каждой паре - координата по горизонтали, второе по вертикали. Координаты отсчитываются от верхнего левого угла. Координаты верхней левой клетки – (1,1). Далее, во входном файле следуют строки, состоящие из нулей и единиц (1обозначает «стенку», т.е. клетку, на которую нельзя наступать), 0 – свободную клетку). Эти строки представляют собой поле лабиринта. Начальная и конечная клетки свободны. Вдоль границ лабиринта все клетки единичные.

Выходные данные

В выходной файл необходимо вывести длину кратчайшего пути из начальной клетки в конечную. Длина пути – это число переходов с клетки на клетку. Если такой путь невозможен, то необходимо вывести символ #.

Примеры

input.txt	*output.txt*

`5 3 4 2 2 2`
`11111`
`10101`
`11111`
	`#`

10 4 2 2 9 3 1111111111 1000010001 1001000001 1111111111	8

Методические указания

Найдите клетки, соседние с начальной, и поместите их в список. Далее, двигаясь по этому списку, найдите клетки, соседние с последними, и поместите их в новый список. И так далее. Как только конечная точка попадет в список, остановитесь. Если в список перестали попадать новые клетки, а конечной клетки среди них нет, значит попасть в конечную клетку невозможно. Подробнее смотрите в Википедии «поиск в ширину».

Задача 8. Проверка графа на ацикличность. Обход графа в глубину.

Ограничение по времени: 1 секунда на тест

В данной задаче задан граф (набор точек-вершин и соединяющих их ребер). Требуется определить, есть ли в этом графе путь (последовательность вершин, соединенных ребрами), такой, что начало и конец пути – одна и та же вершина. При этом каждое ребро в этом пути ранее может повторяться.

Входные данные

Граф во входных данных представлен списком ребер. В первой строчке входного файла располагается число N ($1 \le N \le 1000000$) – количество ребер в графе. Далее располагаются N строк, содержащих по два числа X и Y – номера вершин графа, соединенных ребром ($1 \le X,Y \le 200000$). Граф неориентированный, т.е. можно перейти по ребру как из вершины X в вершину Y, так и наоборот. Любые две вершины в графе могут соединяться только одним ребром. Граф может быть несвязным, т. е. не из каждой вершины можно добраться до любой другой.

Выходные данные

В выходной файл нужно вывести 1, если граф цикличен (в нем есть путь описанного выше вида), и 0, если нет.

Примеры

input.txt	*output.txt*
3 1 2 2 3 3 1	1
2 1 4 2 3	0
5 1 2 2 3 2 4 4 5 4 6	0

Методические указания

Можно начать движение из произвольной вершины графа в любую другую, с которой эта вершина соединена. После перехода пройденное ребро удаляется, и движение продолжается дальше. Если вы попали в тупик, то вернитесь к пройденным вершинам, из которых еще есть не пройденные ребра. Если в процессе движения вы попали в вершину, в которой уже были ранее, то граф содержит цикл. Если закончились ребра, то цикла нет. Подробнее смотрите в Википедии «граф» и «обход графа в глубину» .

Задача 9. Число перестановок. Комбинаторные алгоритмы.

Ограничение по времени: `1 секунда на тест`

Сколько слов можно составить, переставляя буквы заданного слова.

Входные данные

Во входном файле располагается строчка, содержащая заглавные буквы английского алфавита. Длина строки не превышает 255 символов. Входные данные таковы, что результат не превосходит 2147483647.

Выходные данные

В выходной файл требуется вывести число возможных перестановок.

Примеры

input.txt	*output.txt*
BBABCCCAC	1260

Методические указания

В данной задаче вычислить число перестановок можно по формуле, не прибегая к длинной арифметике и перебору. Главное - не допустить арифметического переполнения при получении промежуточных результатов. В данном примере результат можно получить по формуле N=9!/(2! *3!*4!)=1260. Разбейте числитель и знаменатель в формуле на отдельные сомножители и проведите все возможные сокращения. Подробнее смотрите в Википедии «комбинаторика».

Задача 10. Гвоздики. Динамическое программирование .

Ограничение по времени: `1 секунда на тест`

Необходимо определить наименьшую сумму длин ниток, которые нужно привязать к гвоздикам так, чтобы к каждому гвоздику была привязана, по крайней мере, одна нитка и оба конца каждой нитки были привязаны к гвоздикам. Гвоздики забиты в один ряд .

Входные данные

Во входном файле первым располагается число N промежутков между гвоздиками ($1 \le N \le 10000$). Далее следуют N чисел L(k) ($0 \le L(k) \le 100$), каждое из которых является длиной промежутка между соответствующей парой гвоздей.

Выходные данные

В выходной файл требуется вывести наименьшую сумму.

Примеры

input.txt	*output.txt*
5 2 3 4 2 8	14

Методические указания

Если программу записать рекурсивно, т. е. свести задачу к ряду подзадач, тогда, сохраняя полученные ранее результаты в таблице, можно добиться высокой эффективности алгоритма. В данной задаче надо смотреть на то, как привязаны нитки к последним трем гвоздям. Подробнее смотрите в Википедии «Динамическое программирование».

Задача 11. Игра Калах. Метод ветвей и границ . Отсечения

Ограничение по времени: `5 секунда на тест`

В данной задаче вам предлагается написать «искусственный интеллект» для игры Калах. Подробнее смотрите в Википедии «Калах». Цель данной игры накопить на своем поле (калахе) как можно больше камней. Игровое поле состоит из двух рядов лунок. У каждого игрока по 6 лунок. Лунки соперников расположены напротив друг друга. Также у каждого игрока есть специальная лунка «калах», которая находится справа от других его лунок. В начале игры в лунки помещаются камни. Есть варианты игры, когда в ячейку изначально кладётся 4, 6 или 12 камней. Игроки ходят по очереди, кроме случаев, описанных ниже. В свой ход игрок выбирает одну из своих лунок и

раскладывает все камни из этой лунки в другие лунки, двигаясь против часовой стрелки (слева-направо в своих ячейках и справа-налево в ячейках противника). В каждую лунку он кладет по одному камню, при этом пропуская «калах» противника, но не пропуская свой. Если последний камень игрок кладет в свой калах, то он делает следующий ход. Если последний камень игрок кладет в любую другую лунку, то ход переходит к оппоненту. Если последний камень игрок кладет в свою пустую лунку, то он забирает все камни из противоположной лунки ряда партнера и свой камень впридачу, и кладет их все в свой «калах». Когда очередной игрок не может сделать ход, так как все его лунки пусты — игра заканчивается. Второй игрок после этого переносит все оставшиеся в его лунках камни в свой «калах». После этого проводится подсчет камней и тому, кто набрал больше камней, присуждается победа.

По окончанию срока по задаче 11 будет проведено соревнование между вашими программами среди участников олимпиады. Соревнование будет проводиться по круговой системе, когда каждая программа сыграет с каждой два раза (за первого и за второго игрока). За победу будет присуждаться 2 очка, за ничью 1. При равенстве очков будет учитываться разница камней. По регламенту, если ваша программа не выдаст ход за 5 секунд или выведет «для хода» номер лунки, в которой нет камней, то ей будет засчитываться техническое поражение и все оставшиеся камни на игровом поле достанутся вашему противнику. Указание к решению: рекомендуется в процессе «размышлений» вашей программы выводить промежуточные результаты в выходной файл и менять их по мере обнаружения более лучшего (оптимального) хода. За временем ваша программа должна следить сама.

Входные данные

Во входном файле располагаются 14 чисел – число камней в лунках против часовой стрелки. Первые 6 чисел - это камни в ваших лунках, далее следует число камней в вашем «калахе», затем 6 чисел – количества камней в лунках противника, последнее число – количество камней в «калахе» противника.

Выходные данные

В выходной файл требуется вывести одно число от 1 до 6 – номер лунки, из которой ваша программа считает, что ход будет наилучшим. Нумерация лунок идет слева направо. В указанном ниже примере во второй вашей лунке лежит 5 камней и если сделать ход из этой лунки, то ход можно будет повторить.

Примеры

input.txt	*output.txt*
4 5 6 2 3 2 0 3 7 11 0 0 8 0	2

Методические указания

Методика решения подобных задач достаточно хорошо проработана. В целом вы должны рекурсивно рассмотреть дерево возможных позиций. За вашего противника следует отобрать ходы, ведущие к ухудшению вашей позиции. Для себя, естественно, отобрать ходы, ведущие к улучшению. Если при данной глубине просмотра ходов конец игры не будет достигаться, то вы должны написать оценочную функцию. Например, в простейшем случае это разность «калахов». Далее глубину просмотра позиции можно увеличить, если применить отсечения. Подробнее смотрите в Википедии «Метод ветвей и границ».