Q1删除单词后缀

描述

给一组各分别以er、ly和ing结尾的单词， 请删除每个单词的结尾的er、ly或ing， 然后按原顺序输出删除后缀后的单词（删除后缀后的单词长度不为0）。

关于输入

输入的第一行是一个整数n（n≤50），表示后面有n个单词；

其后每行一个单词（单词中间没有空格，每个单词最大长度为32）。

关于输出

按原顺序输出删除后缀后的单词。

例子输入

3

referer

lively

going

例子输出

refer

live

go

例程

/\*删除单词后缀\*/

#include<iostream>

#include<cstring>

using namespace std;

int main()

{

int n, len;

char letter[33], \*tail[3] = { "er","ly","ing" };

cin >> n;

while (n--)

{

cin >> letter;

len = strlen(letter);

for(int i=0;i<2;i++)//检验er或ly

if (!strcmp(letter + len - 2, tail[i]))

letter[len - 2] = 0;

if (!strcmp(letter + len - 3, tail[2]))//检验ing

letter[len - 3] = 0;

cout << letter << endl;

}

return 0;

}

Q2不与最大数相同的数字之和

描述

输出一个整数数列中不与最大数相同的数字之和

关于输入

输入分为两行：

第一行为N(N为接下来数字的个数，N<=100)；

第二行为N个整数，以空格隔开；

关于输出

输出为N个数中除去最大数其余数字之和。（注意，最大数可能出现多次）

例子输入

3

1 2 3

例子输出

3

例程

/\*不与最大数相同的数字之和\*/

#include<iostream>

using namespace std;

int main()

{

int n, num[100], max = -2147483647, sum = 0;

cin >> n;

for (int i = 0;i < n;i++)

{

cin >> num[i];

max = max > num[i] ? max : num[i];

}

for (int i = 0;i < n;i++)

if (num[i] - max)

sum += num[i];

cout << sum << endl;

return 0;

}

Q3区间合并

描述

给定 n 个闭区间 [ai; bi]，其中i=1,2,...,n。 这些区间可以用一组不间断的闭区间表示。我们的任务是找出这些区间是否可以用一个不间断的闭区间表示，如果可以的话将这个最小的闭区间输出，否则输出no。

关于输入

第一行为一个整数n，3 ≤ n ≤ 50000。表示输入区间的数量。在第i+1行上（1 ≤ i ≤ n），为两个整数 ai 和 bi ，整数之间用一个空格分隔，表示区间 [ai; bi]（其中 1 ≤ ai ≤ bi ≤ 10000）。

关于输出

输出为一行，如果这些区间可以合并为一个闭区间，输出这个闭区间，否则输出”no”

例子输入

5

5 6

1 5

10 10

6 9

8 10

例子输出

1 10

例程

/\*区间合并\*/

#include<iostream>

using namespace std;

int main()

{

int n;

cin >> n;

struct room

{

int left, right;

}r[50000];

room \*num[50000] = { NULL }, \*temp = NULL;

for (int i = 0;i < n;i++)

{

num[i] = &r[i];

cin >> r[i].left >> r[i].right;

}

for (int i = 0;i < n - 1;i++)//按照左值冒泡

for (int j = 0;j < n - i - 1;j++)

if (num[j]->left > num[j + 1]->left)//交换指针

{

temp = num[j];

num[j] = num[j + 1];

num[j + 1] = temp;

}

int right = num[0]->right;

for (int i = 0;i < n - 1;i++)

{

if (num[i + 1]->left > right)//连不上了

{

cout << "no" << endl;

return 0;

}

else

right = right > num[i + 1]->right ? right : num[i + 1]->right;

}

cout << num[0]->left << ' ' << right << endl;

return 0;

}

Q4汽车限行

描述

为了缓解交通压力、减少空气污染，B市市政府决定在工作日（周一至周五）对机动车进行限行，每辆机动车每周将有一个工作日不能出行，但周末不限行。假设该政策从2000年1月1日起开始执行。限行安排为：

尾号为1和6：周一限行

尾号为2和7：周二限行

尾号为3和8：周三限行

尾号为4和9：周四限行

尾号为5、0和字母：周五限行

已知2000年1月1日为周六，现在给出一些日期和车牌号，求问该机动车在该天是否限行。

关于输入

输入第一行为一个整数n（1<=n<=100），表示测试的组数。

之后输入n行，每一行由一个日期和一个车牌号组成，用空格分隔。日期的格式为“YYYY-MM-DD"，例如“2000-01-23”，日期保证在2000年1月1日～9999年12月31日之间；车牌号的格式为6位字符串，由数字和大写字母组成，例如“A123Q6”。

关于输出

输出共n行，对于每一组测试输出一行。如果给出的车辆在给出日期限行，则输出“yes”，否则输出“no”。

例子输入

10

2016-01-04 8WNVKN

4592-11-05 JAR2RU

2470-04-15 O54FSL

7730-08-14 WDI3DO

9895-01-09 1CQZ8N

6293-10-20 V45WMK

5304-10-11 RTDY7U

9810-07-14 NAC5LS

7268-11-07 PWQZ1M

6140-09-20 HJR1QV

例子输出

no

no

no

no

no

yes

no

no

no

no

例程：

/\*汽车限行\*/

#include<iostream>

using namespace std;

int main()

{

int n, year, month, day, week, mon[13] = { 0,31,28,31,30,31,30,31,31,30,31,30,31 }, leap;

char tail[7];

cin >> n;

while (n--)

{

cin >> year;

cin.get();

cin >> month;

cin.get();

cin >> day;

cin >> tail;

week = 5;

for (int i = 2000;i < year;i++)

{

leap = (i % 4 == 0) && (i % 100 != 0) || (i % 400 == 0);

week += 365 + leap;

}

leap = (year % 4 == 0) && (year % 100 != 0) || (year % 400 == 0);

for (int i = 1;i < month;i++)

week += mon[i];

if (month > 2)

week += leap;

week += day;//总天数

week %= 7;

if (week == 6 || week == 0)//周末不限行

{

cout << "no" << endl;

continue;

}

if (!((tail[5] - '0') % 5) || tail[5] >= 'A' && tail[5] <= 'Z')//周五

{

if (week == 5)

cout << "yes";

else

cout << "no";

cout << endl;

continue;

}

if ((tail[5] - '0') % 5 == week)

cout << "yes";

else

cout << "no";

cout << endl;

}

return 0;

}

Q5基因检测

描述

用一个字符串表示一段基因，例如：“CTATGGGTTT”。两段基因的相似度定义为它们所包含的最大公共子串的长度。例如：“CCTTGG”和“TGGGC”的最大公共子串为“TGG”，它的长度为3，则我们称“CCTTGG”和“TGGGC”的相似度为3。现给定两段基因，要求计算它们的相似度。

关于输入

输入第一行包含一个正整数N（0

关于输出

对于每组测试数据输出一行，该行包含一个整数，表示给定基因段的相似度。

例子输入

2

CCCCC TTTTTGGGGGCC

ACTGGG DDD

例子输出

2

0

例程：

/\*基因检测\*/

#include<iostream>

#include<cstring>

using namespace std;

int main()

{

int n, lena, lenb;

char a[100], b[100];

cin >> n;

while (n--)

{

cin >> a >> b;

lena = strlen(a);

lenb = strlen(b);

int flag = 0;

int len = lena < lenb ? lena : lenb;

while (len)

{

for (int sta = 0;sta + len <= lena && !flag;sta++)//移动a起始位置

for (int stb = 0;stb + len <= lenb && !flag;stb++)//移动b起始位置

if (!strncmp(a + sta, b + stb, len))

flag++;

if (flag)

break;

len--;

}

cout << len << endl;

}

return 0;

}

Q6左手定则

玩过RPG（尤其是国产RPG）的童鞋都应该对迷宫这种神棍的设定深恶痛绝，尤其是当你转了半小时之后发现回到了原地，这种感觉真是无比的痛苦。。。万一游戏还中途崩溃了那真是连掀桌子、砸键盘、摔鼠标的心都有了……

经过无数次的TRIAL AND ERROR之后，玩家终于下定决心认定迷宫存在的意义就是延长游戏时间，SO，他决定借鉴著名的左手定则（就是在每一个路口，我们都选择最左边的方向，左转的优先级最高，其次为向前，最后为右转，如果实在走进了一个死胡同，那就连续右转两次，回头向后走。稍微研究一下这种走迷宫的方法，我们就发现在这个过程中，事实上我们的左手可以始终放在墙上。）对迷宫进行探索。

但是呢，左手定则并不能保证遍历到迷宫中的每一个点。悲剧的是，某项重要的通关道具被放在了这个迷宫中……幸运的是，游戏迷宫的地图可以绘制出来，现在请求你的帮助。对于一个给定的地图，他想知道是不是能够通过左手定则取得这件道具。

关于输入

多组数据。

对于每组数据，第一行有两个整数N,M代表接下来有n行字符串，每行m个字符，其中0其中‘#’表示墙，‘S’表示起点，‘T’表示道具，‘.’表示空地。

接下来是一个方向（NSWE），表示起始面向的方向。

数据保证最外一圈都是墙。

关于输出

对于每组数据输出一行。‘YES’表示可以到达，‘NO’表示无法到达。

例子输入

8 10

##########

#...T....#

#.####...#

#.#..#.#.#

#.#....#.#

#.####.#.#

#......#S#

##########

N

8 10

##########

#........#

#.####...#

#.#T.#.#.#

#.#....#.#

#.####.#.#

#......#S#

##########

N

8 10

##########

#....#...#

#..#.#...#

#..#.....#

#..#.#S###

#..#.#...#

#....#T..#

##########

N

例子输出

YES

NO

YES

例程：

/\*左手定则\*/

#include<iostream>

using namespace std;

const int Max = 100;

bool maze[Max][Max], flag[Max][Max][4];

int sx, sy, xx, yy, direct, dx[4] = { 0,-1,0,1 }, dy[4] = { -1,0,1,0 };

bool demaze(int x, int y, int dir)

{

if (flag[x][y][dir])//以同样的方向来过一次

return 0;

flag[x][y][dir] = 1;

if (x == xx && y == yy)//找到道具

return 1;

int l\_r = (dir + 3) % 4;//左手的方向

if (!maze[x + dx[l\_r]][y + dy[l\_r]])

return demaze(x + dx[l\_r], y + dy[l\_r],l\_r);

else if (!maze[x + dx[dir]][y + dy[dir]])//前方

return demaze(x + dx[dir], y + dy[dir], dir);

else

{

l\_r = (dir + 1) % 4;//右手的方向

if (!maze[x + dx[l\_r]][y + dy[l\_r]])

return demaze(x + dx[l\_r], y + dy[l\_r], l\_r);

else

{

l\_r = (dir + 2) % 4;//后背的方向

return demaze(x, y, l\_r);

}

}

}

int main()

{

int m, n;

char temp;

while (cin >> m >> n)

{

memset(maze, 0, sizeof(maze));

memset(flag, 0, sizeof(flag));

for (int i = 0;i < m;i++)

for (int j = 0;j < n;j++)

{

cin >> temp;

if (temp == '#')

maze[i][j] = 1;

else if (temp == 'S')

{

sx = i;

sy = j;

}

else if (temp == 'T')

{

xx = i;

yy = j;

}

}

cin >> temp;

switch (temp)

{

case'W':direct = 0;break;

case'N':direct = 1;break;

case'E':direct = 2;break;

case'S':direct = 3;break;

}

if (demaze(sx, sy, direct))

cout << "YES" << endl;

else

cout << "NO" << endl;

}

return 0;

}

Q7多边形游戏

描述

一个多边形，开始有n个顶点。每个顶点被赋予一个正整数值，每条边被赋予一个运算符“+”或“\*”。所有边依次用整数从1到n编号。

现在来玩一个游戏，该游戏共有n步：

第1步，选择一条边，将其删除

随后n-1步，每一步都按以下方式操作：（1）选择一条边E以及由E连接着的2个顶点v1和v2； (2）用一个新的顶点取代边E以及由E连接着的2个顶点v1和v2，将顶点v1和v2的整数值通过边E上的运算得到的结果值赋给新顶点。

最后，所有边都被删除，只剩一个顶点，游戏结束。游戏得分就是所剩顶点上的整数值。那么这个整数值最大为多少？

关于输入

第一行为多边形的顶点数n（n≤20），其后有n行，每行为一个整数和一个字符，整数为顶点上的正整数值，字符为该顶点到下一个顶点间连边上的运算符“+”或“\*”（最后一个字符为最后一个顶点到第一个顶点间连边上的运算符）。

关于输出

输出仅一个整数，即游戏所计算出的最大值。

例子输入

4

4 \*

5 +

5 +

3 +

例子输出

70

提示

小规模问题可不必用动态规划方法编程求解，仅用递归就可以求解。

计算中不必考虑计算结果超出整数表达范围的问题，给出的数据能保证计算结果的有效性。

在给的例子中，计算过程为(3+4)\*(5+5)=70。

例程

/\*多边形游戏\*/

#include<iostream>

using namespace std;

int num[20], n;

char cal[21];

int f(int start, int end)//从start到end的最大值

{

if (start == end)

return num[start % n];

int max = 0;

for (int i = start;i < end;i++)//枚举所有分成两个部分的情况

{

int temp;

switch (cal[i % n])

{

case '+':temp = f(start, i) + f(i + 1, end);break;

case '\*':temp = f(start, i) \* f(i + 1, end);break;

default:cout << "error";break;

}

max = max > temp ? max : temp;

}

return max;

}

int main()

{

int max = 0, temp;

cin >> n;

for (int i = 0; i < n; i++)

cin >> num[i] >> cal[i];

for (int i = 0; i < n; i++)

{

temp = f(i, i + n - 1);

max = max > temp ? max : temp;

}

cout << max << endl;

return 0;

}