Problem A
サイゾウ

Time Limit: 3 sec

東京大学プログラミングコンテスト
31 May 2008

ある国で「サイゾウ」というテレビ番組が流行している．この番組は参加者がフィールドアスレチックに挑戦し，見事攻略すれば賞金をもらえるというものである．

フィールドアスレチックは高さが異なるブロックを一列に並べて作られていて，攻略のためにはいかにして段差を登り降りするかが重要になる（図 1）．この番組に参加することになったあなたの友人は，フィールドアスレチックの構造が与えられたときに，登らなければならない最大の段差と降りなければならない最大の段差を計算するプログラムを書いてほしいと，凄腕プログラマーであるあなたに依頼してきた．

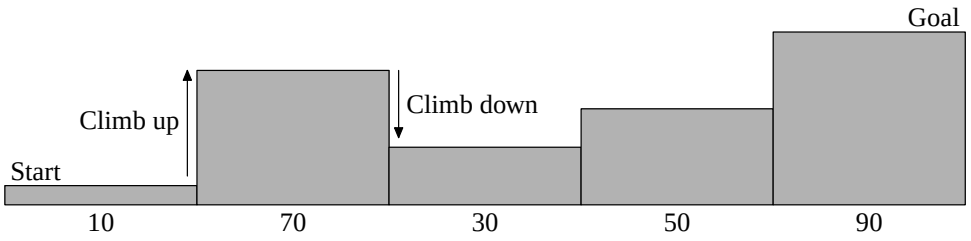


図 1　アスレチックの構造の例（入力例の最初のデータセット）．

Input

入力の 1 行目にはデータセットの個数 t ($0 < t \leq 100$) が与えられる．この行に引き続き t 個のデータセットが与えられる．

データセットの 1 行目はフィールドアスレチックを構成しているブロックの個数 n ($2 \leq n \leq 100$) である．2 行目にはスタートからゴールまでのブロックの高さを示す整数が順に n 個与えられる．1 番目がスタート， n 番目がゴールに対応している．これらの整数は 1 つの空白文字で区切られている．各ブロックの高さ h は $0 < h \leq 1000$ を満たす．

Output

各データセットに対し，登らなければならない最大の段差の大きさと，降りなければならない最大の段差の大きさを，1 つの空白文字で区切って 1 行に出力せよ．なお，登る段差がない，または降りる段差がない場合は，対応する最大の段差の大きさは 0 とすること．

Sample Input

```
5
5
10 70 30 50 90
2
20 100
2
100 30
3
50 50 50
7
123 45 678 901 234 567 890
```

Output for the Sample Input

```
60 40
80 0
0 70
0 0
633 667
```

Problemsetter: Yoshitake Matsumoto

Problem B
完全数

Time Limit: 3 sec

東京大学プログラミングコンテスト
31 May 2008

ある整数 N に対し，その数自身を除く約数の和を S とする．$N = S$ のとき N は完全数 (perfect number)，$N > S$ のとき N は不足数 (deficient number)，$N < S$ のとき N は過剰数 (abundant number) と呼ばれる．与えられた整数が，完全数・不足数・過剰数のどれであるかを判定するプログラムを作成せよ．

プログラムの実行時間が制限時間を越えないように注意すること．

Input

入力はデータセットの並びからなる．データセットの数は 100 以下である．

各データセットは整数 N ($0 < N \leq 100000000$) のみを含む 1 行からなる．

最後のデータセットの後に，入力の終わりを示す 0 と書かれた 1 行がある．

Output

各データセットに対し，整数 N が完全数ならば “`perfect number`”，不足数ならば “`deficient number`”，過剰数ならば “`abundant number`” という文字列を 1 行に出力せよ．

Sample Input

```
1
2
3
4
6
12
16
28
33550336
99999998
99999999
100000000
0
```

Output for the Sample Input

```
deficient number
deficient number
deficient number
deficient number
perfect number
abundant number
deficient number
perfect number
perfect number
deficient number
deficient number
abundant number
```

Problemsetter: Yoshitake Matsumoto

Problem C
ラミー

Time Limit: 5 sec

東京大学プログラミングコンテスト
31 May 2008

あなたの友達は最近 UT-Rummy というカードゲームを思いついた．

このゲームで使うカードには赤・緑・青のいずれかの色と 1 から 9 までのいずれかの番号がつけられている．このゲームのプレイヤーはそれぞれ 9 枚の手札を持ち，自分のターンに手札から 1 枚選んで捨てて，代わりに山札から 1 枚引いてくるということを繰り返す．このように順番にターンを進めていき，最初に手持ちのカードに 3 枚ずつ 3 つの「セット」を作ったプレイヤーが勝ちとなる．セットとは，同じ色の 3 枚のカードからなる組で，すべて同じ数を持っているか連番をなしているもののことを言う．連番に関しては，番号の巡回は認められない．例えば，7, 8, 9 は連番であるが 9, 1, 2 は連番ではない．

あなたの友達はこのゲームをコンピュータゲームとして売り出すという計画を立てて，その一環としてあなたに勝利条件の判定部分を作成して欲しいと頼んできた．あなたの仕事は，手札が勝利条件を満たしているかどうかを判定するプログラムを書くことである．

Input

入力の 1 行目にはデータセット数を表す数 T ($0 < T \leq 50$) が与えられる．この行に引き続き T 個のデータセットが与えられる．

各データセットは 2 行からなり，1 行目には i 番目 ($i = 1, 2, \ldots, 9$) のカードの数字 n_i がそれぞれスペースで区切られて与えられ，2 行目には i 番目のカードの色を表す文字 c_i がそれぞれスペースで区切られて与えられる．カードの数字 n_i は $1 \leq n_i \leq 9$ を満たす整数であり，カードの色 c_i はそれぞれ “R”, “G”, “B” のいずれかの文字である．

1 つのデータセット内に色と数字がともに同じカードが 5 枚以上出現することはない．

Output

各データセットにつき，勝利条件を満たしていれば “1”，そうでなければ “0” と出力せよ．

Sample Input

```
5
1 2 3 3 4 5 7 7 7
R R R R R R G G G
1 2 2 3 4 4 4 4 5
R R R R R R R R R
1 2 3 4 4 4 5 5 5
R R B R R R R R R
1 1 1 3 4 5 6 6 6
R R B G G G R R R
2 2 2 3 3 3 1 1 1
R G B R G B R G B
```

Output for the Sample Input

```
1
0
0
0
1
```

Problemsetter: Masashi Tsubosaka

Problem D
バトルタウン

Time Limit: 3 sec

東京大学プログラミングコンテスト
31 May 2008

あなたは友人たちとゲームを開発することにした．ゲームのタイトルは「バトルタウン」．戦車による市街戦をテーマにしたゲームだ．ゲーム開発の第一歩として，次のようなプロトタイプを開発することにした．

このプロトタイプでは，登場する戦車はプレイヤーの戦車のみであり，敵の戦車は登場しない．プレイヤーの戦車はプレイヤーの入力に従ってマップ上で様々な動作をする．以下に，戦車の動作に関する仕様を述べる．

マップの構成要素は表 1 の通りである．戦車が移動できるのはマップ内の平地の上だけである．

表 1　マップの構成要素

文字	要素
.	平地
*	レンガの壁
#	鉄の壁
-	水
^	戦車（上向き）
v	戦車（下向き）
<	戦車（左向き）
>	戦車（右向き）

プレイヤーの入力は文字の列で与えられる．各文字に対応する動作は表 2 の通りである．

表 2　プレイヤーの入力に対する動作

文字	動作
U	Up: 戦車を上向きに方向転換し，さらに一つ上のマスが平地ならばそのマスに移動する
D	Down: 戦車を下向きに方向転換し，さらに一つ下のマスが平地ならばそのマスに移動する
L	Left: 戦車を左向きに方向転換し，さらに一つ左のマスが平地ならばそのマスに移動する
R	Right: 戦車を右向きに方向転換し，さらに一つ右のマスが平地ならばそのマスに移動する
S	Shoot: 戦車が現在向いている方向に砲弾を発射する

砲弾はレンガの壁あるいは鉄の壁にぶつかるか，マップの外に出るまで直進する．レンガの壁にぶつかった場合は，砲弾は消滅し，レンガの壁は平地に変化する．鉄の壁にぶつかった場合は，砲弾は消滅するが，鉄の壁は変化しない．マップの外に出たときは砲弾はそのまま消滅する．

あなたの仕事は，初期マップとプレイヤーの入力操作列が与えられたときに，最終的なマップの状態を出力するプログラムを作成することである．

Input

入力の 1 行目にはデータセット数を表す数 T ($0 < T \leq 100$) が与えられる．この行に引き続き T 個のデータセットが与えられる．

各データセットの 1 行目にはマップの高さと幅を表す整数 H と W が 1 つの空白文字で区切られて与えられる．これらの整数は $2 \leq H \leq 20, 2 \leq W \leq 20$ を満たす．続く H 行はマップの初期状態を表す．各行は長さ W の文字列である．マップに含まれる文字はすべて表 1 で定義されたものであり，全体でちょうど 1 つの戦車を含む．マップに続く行には入力操作列の長さ N ($0 < N \leq 100$) が与えられ，次の行には入力操作列を表す長さ N の文字列が与えられる．入力操作列は表 2 で定義された文字のみからなる文字列である．

Output

各データセットに対し，すべての入力操作を終えた後のマップを，入力マップと同じ形式で出力せよ．データセットの間には 1 行の空行を出力せよ．最後のデータセットの後に余計な空行を出力しないよう注意すること．

Sample Input

```
4
4 6
*.*..*
*.....
..-...
^.*#..
10
SRSSRRUSSR
2 2
<.
..
12
DDSRRSUUSLLS
3 5
>-#**
.-*#*
.-**#
15
SSSDRSSSDRSSSUU
5 5
v****
*****
*****
*****
*****
44
SSSSDDRSDRSRDSULUURSSSSRRRRDSSSSDDLSDLSDLSSD
```

Output for the Sample Input

```
*....*
......
..-...
..>#..

<.
..

^-#**
.-.#*
.-..#

.....
.***.
..*..
..*..
....v
```

Problemsetter: Yoshitake Matsumoto

Problem E
カントリーロード

Time Limit: 5 sec

東京大学プログラミングコンテスト
31 May 2008

ある過疎地域での話である．この地域ではカントリーロードと呼ばれるまっすぐな道に沿って，家がまばらに建っている．今までこの地域には電気が通っていなかったのだが，今回政府からいくつかの発電機が与えられることになった．発電機は好きなところに設置できるが，家に電気が供給されるにはどれかの発電機に電線を介してつながっていなければならず，電線には長さに比例するコストが発生する．地域で唯一の技術系公務員であるあなたの仕事は，すべての家に電気が供給されるという条件の下で，できるだけ電線の長さの総計が短くなるような発電機および電線の配置を求めることである．なお，発電機の容量は十分に大きいので，いくらでも多くの家に電気を供給することができるものとする．

サンプル入力の 1 番目のデータセットを図 2 に示す．この問題に対する最適な配置を与えるには，図のように $x = 20$ と $x = 80$ の位置に発電機を配置し，それぞれ図中のグレーで示した位置に電線を引けばよい．

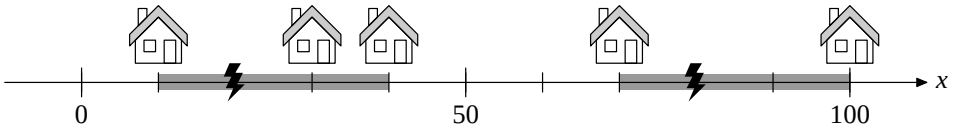


図 2　発電機と電線の配置の例（入力例の最初のデータセット）．

Input

入力の 1 行目にはデータセットの個数 t ($0 < t \leq 50$) が与えられる．

引き続き t 個のデータセットが与えられる．データセットはそれぞれ次のような形式で 2 行で与えられる．

n k
x_1 x_2 ... x_n

n は家の戸数，k は発電機の個数である．$x_1, x_2, \ldots, x_n$ はそれぞれ家の位置を表す一次元座標である．これらの値はすべて整数であり，$0 < n \leq 100000$, $0 < k \leq 100000$, $0 \leq x_1 < x_2 < \ldots < x_n \leq 1000000$ を満たす．

さらに，データセットのうち 90% は，$0 < n \leq 100$, $0 < k \leq 100$ を満たしている．

Output

各データセットに対し，必要な電線の長さの総計の最小値を 1 行に出力せよ．

Sample Input

```
6
5 2
10 30 40 70 100
7 3
3 6 10 17 21 26 28
1 1
100
2 1
0 1000000
3 5
30 70 150
6 4
0 10 20 30 40 50
```

Output for the Sample Input

```
60
13
0
1000000
0
20
```

Problemsetter: Yoshitake Matsumoto

Problem F
リズムマシーン

Time Limit: 5 sec

東京大学プログラミングコンテスト
31 May 2008

Advanced Computer Music 社（ACM 社）は，あらかじめプログラムされたリズム通りに音楽を演奏するリズムマシーンを販売していた．ある時，ACM 社は新しいリズムマシーンを開発して売り出そうとしていた．ACM 社の旧製品は同時に 1 つの音しか鳴らすことができなかったのに対し，新製品では最大で 8 つの音を同時に鳴らせるようになるというのが一番の目玉機能であった．今まで複数の旧製品を利用して演奏する必要のあった曲が新製品 1 台で済むようになるので，ACM 社は新製品への移行を推進させるために，複数の旧製品向けのリズムパターンを 1 つの新製品向けのリズムパターンに変換するプログラムを作ることにした．

ACM 社のリズムマシーンでは，同時にどの音を鳴らすかを 2 桁の 16 進数で表現する．ACM 社のリズムマシーンは 8 つの異なる音を鳴らすことが可能で，それぞれの音には 0 から 7 の番号が割り当てられている．あるタイミングにおいて，音 i $(0 \le i < 8)$ を鳴らす場合を $s_i = 1$，鳴らさない場合を $s_i = 0$ とする．このとき，それぞれの音を同時に鳴らしたような和音を $S = \sum_i \left(s_i \times 2^i\right)$ という値で表し，この値を 2 桁の 16 進数表記で表した「和音表現」がリズムパターンの中で用いられる（16 進数の英字は大文字を用いる）．例えば，音 0, 6, 7 を同時に鳴らすような和音は $S = 2^0 + 2^6 + 2^7 = \mathrm{C1}_{(16)}$ となるから “C1” と表現され，また何も鳴らさないような「和音」は “00” と表現される．

リズムパターンは，上記のような和音表現を 1 つ以上並べたものとして与えられる．あるリズムパターン文字列は，1 小節内の演奏パターンを示している．それぞれの和音を鳴らすタイミングを小節内の相対位置 t $(0 \le t < 1)$ で表現することにする．k 個の和音表現からなるリズムパターン文字列は，小節を k 等分しそれぞれの和音を順に $t = 0/k, 1/k, \ldots, (k-1)/k$ のタイミングで演奏するようなリズムパターンを表している．例えば，リズムパターン “01000003” は，$t = 0/4$ のタイミングで音 0 を演奏し，$t = 3/4$ のタイミングで音 0, 1 を演奏することを表す．また，リズムパターン “00” は小節内で何も音を鳴らさないことを表す（リズムパターンには和音表現が 1 つ以上必要であることに注意せよ）．

旧製品は同時に 1 つの音しか鳴らせないため，旧製品向けのリズムパターン文字列内には “00”, “01”, “02”, “04”, “08”, “10”, “20”, “40”, “80” のいずれかの和音表現しか現れない．旧製品向けのリズムパターンを N 個 $(1 \le N \le 8)$ 受け取り，それらのリズムパターンを同時に演奏するような新製品向けのリズムパターンを出力するプログラムを書いて欲しい．

与えられる N 個のリズムパターンにおいて，まったく同じタイミングで同じ音が演奏されることはないと仮定してよい．

Input

最初の行にデータセットの数が与えられる．次の行以降には，それぞれのデータセットが順に記述されている．データセットの数は 120 を越えないと仮定してよい．

それぞれのデータセットは以下のような形式で与えられる．

N
R_1
R_2
$\ldots$
R_N

R_i $(1 \leq i \leq N)$ はそれぞれ旧製品向けのリズムパターンである．

各リズムパターンは最大で 2048 文字（1024 和音表現）である．与えられるリズムパターンは必ずしも最短の表現になっていないことに注意せよ．

Output

各データセットについて，与えられた N 個のリズムパターンをすべて同時に演奏するような最短のリズムパターンを生成し，1 行で出力せよ．そのようなリズムパターンが 2048 文字を越える場合，リズムパターンの代わりに “`Too complex.`” という文字列を出力せよ．

Sample Input

```
5
2
01000100
00020202
2
0102
00000810
1
0200020008000200
5
0001
000001
0000000001
00000000000001
00000000000000000001
1
000000
```

Output for the Sample Input

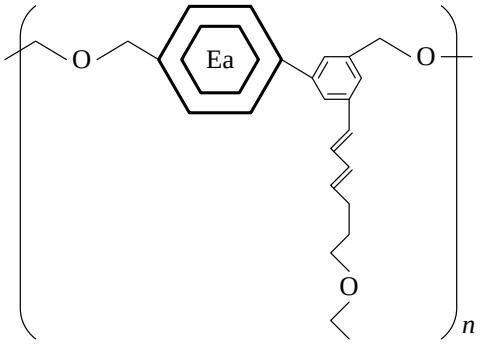
```
01020302
01000A10
02020802
Too complex.
00
```

Problemsetter: Yuta Kitamura

Problem G
エナジー・トランスポーター

東京大学プログラミングコンテスト
Time Limit: 5 sec
31 May 2008

とある研究所で，エネルギー伝達用の媒体の開発をしていた．この媒体は図 3 に示すような特殊な物質からなるポリマー構造をなしている．



$(-\alpha\text{–Ea–}\beta\text{–})_n$

図 3　エネルギー伝達用媒体の構造.

図の Ea で示した部分がこの媒体のもっとも特徴的な部位のエナジーアキュムレータ (Energy Accumulator) である．この Ea 基は 1 kJ 幅で離散化された多様なエネルギー状態を取ることができる．ある Ea 基を励起させると，その Ea 基の α 側に結合している隣接した Ea 基に蓄積されている全エネルギーを β 側に結合している隣接した Ea 基に移動させるような効果を持つ発熱反応が引き起こされる（図 4）．この反応の際，励起される Ea 基のエネルギーが 1 kJ 消費される．なお，ポリマーの両端に位置する Ea 基やエネルギー状態が 0 kJ になっている Ea 基に対しては励起反応は発生しないこと，および Ea 基は十分に大きなエネルギーを蓄えることが可能であることが知られている．

Before excitation	a kJ	b kJ	c kJ
After excitation	0 kJ	$b - 1$ kJ	$a + c$ kJ

図 4　中央の Ea 基を励起させたときの反応.

この性質を利用することでエネルギーの伝達を可能にしようと考えていたのだが，エネルギーを効率よく伝達するには各 Ea 基を励起させる順番が重要であることに研究者たちは気がついたのである．

幸い，励起させる順番や回数は任意に制御できるのだが，彼らには最適な励起手順がわからない．そこで彼らの発想の足がかりとして，初期状態のエネルギー分布に対して最右 Ea 基（β 末端からもっとも近い Ea 基）に蓄えられうる最大のエネルギー量を計算してもらいたい．

Input

入力は複数のデータセットから構成され，以下のような形式で与えられる．

N
C_1
C_2
$\ldots$
C_N

入力の先頭の整数 N $(0 < N \leq 60)$ が取り扱う問題のデータセット数であり，その後ろ $2N$ 行に渡って，それぞれのデータセットごとの情報 C_k が与えられる．

それぞれのデータセット C_k は以下のような形式で 2 行に渡り与えられる．

L
$E_1\ E_2\ \ldots\ E_L$

L はそれぞれのデータセットで取り扱う媒体の Ea 鎖の長さであり，ここで与えられた数だけ Ea 基が直列に結合していることを意味している．その次の行の L 個の整数 E_k は，長さ L の Ea 鎖のうち α 末端を左端に据えたときに左から数えて k 番目の Ea 鎖にはじめに蓄積されているエネルギー量を kJ 単位で示したものである．

ここで，$0 \leq E_k \leq 4, 1 \leq L \leq 80$ であることが保証されている．

Output

出力は各データセットごとに，与えられた状況下での右端 Ea 鎖に到達可能な最大エネルギーを kJ 単位で，整数値のみを 1 行で記述すること．

Sample Input

```
7
1
2
2
1 2
3
4 1 4
3
4 0 4
5
4 1 4 0 4
5
4 1 4 1 4
5
4 2 4 0 4
```

Output for the Sample Input

```
2
2
8
4
7
12
11
```

Problemsetter: Koji Kojima

Problem H
いけるかな？

Time Limit: 5 sec

東京大学プログラミングコンテスト
31 May 2008

あるところに，日本各地をまわりながら商売をする社長がいた．彼はある日，不思議な切符を手に入れた．その切符を使うと，なんと目的地までの距離によらず電車の運賃が無料になるという．ただし，現在の駅から隣接する駅へ移動するのを 1 ステップと数えたときに，移動するステップ数がちょうど切符に書かれた数と等しくならないと追加料金を取られてしまう．ある区間をただちに折り返すような移動は禁止されているが，既に訪れた駅や区間を複数回通ること自体は許される．たとえば，駅 1・駅 2・駅 1 と移動することはできないが，駅 1・駅 2・駅 3・駅 1・駅 2 のような移動は問題ない．また，最終的に目的地に到着するならば，出発地や目的地も何度でも通ってよい．

社長はさっそくこの切符を次の目的地に行くために使ってみようと考えた．しかし路線図は入り組んでいるため，簡単には経路が定まらない．あなたの仕事は，社長に代わって目的地に無料で到達できるかどうかを判定するプログラムを書くことである．

駅は 1 から N までの番号が振られており，出発地の駅は 1，目的地の駅は N と決まっている．路線図は 2 つの駅を結ぶ区間の列によって与えられる．区間はどちらの方向にも通行することができる．同じ駅同士を結ぶような区間は存在しないことと，ある駅の対を結ぶ区間はたかだか 1 つであることが保証されている．

Input

入力は複数のデータセットからなる．

それぞれのデータセットは次のような形式で与えられる．

N M Z
s_1 d_1
s_2 d_2
...
s_M d_M

N は駅の総数，M は区間の総数であり，Z は切符に書かれたステップ数である．s_i と d_i $(1 \leq i \leq M)$ はそれぞれ駅の番号を表す整数であり，駅 s_i と駅 d_i の間に区間が存在することを表現している．

N, M, Z は正の整数であり，次の条件を満たす：$2 \leq N \leq 50, 1 \leq M \leq 50, 0 < Z < 2^{31}$.

最後のデータセットの後ろに，“0 0 0” と書かれた 1 行がある．

データセットの数は 30 を越えない．

Output

それぞれのデータセットに対し，チケットに書かれているステップ数ちょうどで目的地に到達できるならば “yes”，到達できないならば “no” のみを含む 1 行の文字列を出力せよ.

Sample Input

```
2 1 1
1 2
2 1 2
1 2
3 1 2
1 2
8 8 5778
1 2
2 3
2 4
3 5
5 6
6 7
7 8
4 8
8 8 5777
1 2
2 3
2 4
3 5
5 6
6 7
7 8
4 8
0 0 0
```

Output for the Sample Input

```
yes
no
no
yes
no
```

Problemsetter: Takahiro Nakajima

Problem I
Aaron と Bruce

Time Limit: 5 sec

東京大学プログラミングコンテスト
31 May 2008

Aaron は凶悪な犯罪者である．彼は幾度も犯罪を繰り返しながら（万引き 2 回，のぞき 16 回，下着泥棒 256 回，食い逃げ 65,536 回）も，その人並み外れた身体能力を用いて警察の手から逃れ続けてきた．Bruce は警察官である．彼は突出した運動能力は持っていないが，写真撮影が趣味であり，彼の撮った写真が雑誌に載るほどの腕前を持っている．

ある日，Bruce は山奥に写真撮影をしに来た．すると偶然 Aaron のアジトを突き止めてしまった．Bruce が逃げる Aaron を追っていくと，彼らは落とし穴に落ちて古代遺跡の中に迷い込んでしまった．

古代遺跡の中はいくつかの部屋と，部屋どうしを結ぶ通路で構成されている．古代遺跡に M 個の部屋があるとき，遺跡内の部屋にはそれぞれ 0 から $M-1$ までの番号がつけられている．

Aaron は逃げている間に時効を迎えるかもしれないと考えたので，Bruce が最適に移動したときに一番長い間逃げ続けられるように遺跡の中を移動する．Bruce は早く Aaron を写真に収めたいので，Aaron が最適に移動したときに一番早く Bruce を写真に収められるように遺跡の中を移動する．

Aaron と Bruce は順番に行動する．最初は Aaron の番とする．それぞれの順番のとき，隣接している部屋のどれか 1 つに移動，もしくはその場にとどまることができる．Aaron と Bruce が同じ部屋に入ったとき，Bruce は Aaron を撮影するものとする．

Bruce が Aaron を撮影するのにどれくらいの時間がかかるかを求めて欲しい．時間はターン数で表すこととし，Aaron と Bruce がともに 1 回行動し終わったら 1 ターンと数える．

例えば，図 5 のような状況のとき，Aaron は部屋 5 に逃げ込めば Bruce は 4 ターンでたどり着くが，Aaron が部屋 7 に逃げ込めば Bruce はたどり着くのに 5 ターンかかる．Aaron にとっては部屋 7 に逃げ込むことで一番長く逃げ続けられるので答えは 5 ターンとなる．

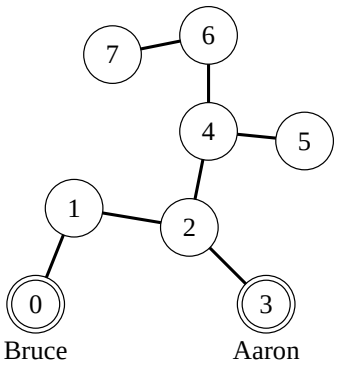


図 5　Aaron が 5 ターン逃げ続けられる例.

また，図 6 のような状況のとき，Aaron が Bruce から遠ざかるように部屋を移動することでいつまでも逃げ回ることができる．

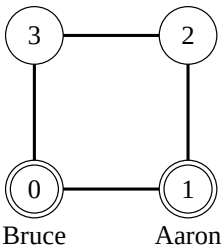


図 6　Aaron がいつまでも逃げ続けられる例.

Input

入力の最初の行にデータセット数を表す数 N ($0 < N \leq 100$) が与えられる．次の行から N 個のデータセットが続く．

各データセットは古代遺跡の形状と Aaron と Bruce の初期位置からなる．初めに，古代遺跡の部屋の数 M ($2 \leq M \leq 50$) が与えられる．次に要素数 $M \times M$ の行列が与えられる．各行列の要素は 0 もしくは 1 であり，i 行目の j 番目の数字が 1 ならば，部屋 i と部屋 j は通路でつながっているということを意味する（ただし，行列の行番号と列番号は 0 から数える）．行列の $(i,\ j)$ 成分と $(j,\ i)$ 成分の値は必ず等しく，$(i,\ i)$ 成分の値は 0 である．続いて，2 つの整数 a, b が与えられる．各整数はそれぞれ Aaron の初期位置の部屋番号 ($0 \leq a < M$) と Bruce の初期位置の部屋番号 ($0 \leq b < M$) を示す．a と b の値は必ず異なる．

Output

各データセットごとに，Bruce が Aaron を撮影するのに何ターンかかるかを 1 行で出力せよ．どれだけたっても撮影できない場合は “`infinity`” と出力せよ．

Sample Input

```
6
8
0 1 0 0 0 0 0 0
1 0 1 0 0 0 0 0
0 1 0 1 1 0 0 0
0 0 1 0 0 0 0 0
0 0 1 0 0 1 1 0
0 0 0 0 1 0 0 0
0 0 0 0 1 0 0 1
0 0 0 0 0 0 1 0
3 0
4
0 1 1 0
1 0 0 1
1 0 0 1
0 1 1 0
1 0
5
0 1 0 1 0
1 0 0 1 0
0 0 0 0 1
1 1 0 0 0
0 0 1 0 0
2 0
5
0 1 0 0 1
1 0 1 1 1
0 1 0 1 0
0 1 1 0 1
1 1 0 1 0
2 4
5
0 1 0 1 0
1 0 1 1 0
0 1 0 0 1
1 1 0 0 1
0 0 1 1 0
3 0
5
0 1 0 1 1
1 0 1 0 1
0 1 0 1 0
1 0 1 0 1
1 1 0 1 0
0 4
```

Output for the Sample Input

```
5
infinity
infinity
2
infinity
1
```

Problemsetter: Ben Hachimori

Problem J
いにしえの数式

Time Limit: 5 sec

東京大学プログラミングコンテスト
31 May 2008

あなたの友人の考古学者が遺跡を発掘していた．ある日，彼は怪しげな記号の羅列が刻まれている石盤を大量に発見した．彼はこの大発見に大喜びし，すぐさま石盤に刻まれている記号の解読を始めた．何週間にもわたる彼の解読の努力の結果，どうやらこれらの石盤には変数・演算子・カッコの組み合わせで成り立っている数式のようなものが刻まれていることがわかった．発掘した石盤と遺跡に関わる文献を調べていくうちに彼はさらに，それぞれの石盤で演算子の結合規則が異なっており，それぞれの石盤の先頭に演算子の結合規則が記されているらしいことを突きとめた．しかし，彼は数学が苦手であるため，演算子の結合規則を見てもすぐには数式がどのように結合しているかがわからなかった．そこで彼は，優れたコンピュータサイエンティストであるあなたに「石盤に書かれている数式がどのように結合しているかを調べてほしい」と依頼してきた．あなたは，彼を手助けすることができるだろうか？

数式中に現れる演算子には優先順位が定められており，優先順位の高い演算子から順に結合していく．同一優先順位をもつ演算子にはそれぞれ結合方向が定められており，左から右に結合する場合は左にある演算子から順に，右から左に結合する場合は右にある演算子から順に結合していく．例えば，`+` より `*` の方が優先順位が高く，`*` が左から右に結合する場合，式 `a+b*c*d` は `(a+((b*c)*d))` のように結合する．サンプル入力にも他の例がいくつか挙げられているので参照のこと．

Input

入力に与えられる数式は，以下の *Expr* で定義されるような文字列である．この定義において，*Var* は変数を表し，*Op* は演算子を表す．

Expr	::=	*Var*
	|	*Expr Op Expr*
	|	“(” *Expr* “)”
Var	::=	“a” | “b” | ... | “z”
Op	::=	“+” | “−” | “*” | “/” | “<” | “>” | “=” | “&” | “|” | “^”

演算子はすべて二項演算子であり，単項演算子やその他の演算子は存在しない．

入力は，いくつかのデータセットからなる．入力の先頭行にデータセット数 D ($D \leq 100$) が与えられ，以降の行に D 個のデータセットが続く．

それぞれのデータセットは，次のようなフォーマットで与えられる．

（演算子の結合規則の定義）
（クエリの定義）

演算子の結合規則の定義は，以下のようなフォーマットで与えられる．この定義において，G は演算子グループの数を表す数である．

G
（演算子グループ 1 の定義）
（演算子グループ 2 の定義）
...
（演算子グループ G の定義）

演算子は，結合の優先順位ごとにグループに分けられている．同一のグループに属する演算子は結合の優先順位が等しく，グループの定義が後に現れる演算子の方が前に現れる演算子より結合の優先順位が高い．

それぞれの演算子グループは，以下のように定義される．

A M p_1 p_2 ... p_M

A は演算子の結合方向を表す文字で，“L” か “R” のいずれかである．“L” は左から右に結合することを表し，“R” は右から左に結合することを表す．M ($M > 0$) は演算子グループに含まれる演算子の数であり，p_i ($1 \leq i \leq M$) はグループに含まれる演算子を表す文字である．同じ演算子がグループ内に 2 度現れることはなく，また，データセット内にグループをまたがって同じ演算子が 2 度現れることもない．

演算子の定義には，上記の *Op* で定義された演算子のみが現れる．1 つのデータセットにおいて，必ず 1 つ以上の演算子が定義されると仮定してよい．

クエリの定義は以下のようなフォーマットで与えられる．

N
e_1
e_2
...
e_N

N ($0 < N \leq 50$) はクエリとして与えられる数式の数である．e_i ($1 \leq i \leq N$) は上記の *Expr* の定義を満たすような式を表す空でない文字列である．e_i の長さはたかだか 100 文字であり，当該データセットで定義されていない演算子は出現しないものと仮定してよい．

Output

データセットごとに，クエリとして与えられたそれぞれの式について，式中の各演算子について必ず 1 つのカッコの組を用いてすべての二項演算を正しく囲み，演算子の結合を表現した文字列を出力せよ．入力の式に現れる変数や演算子の順序を変更してはならない．

それぞれのデータセットの間には空行を出力せよ．出力の末尾に空行を出力してはならない．

Sample Input

```
2
3
R 1 =
L 2 + -
L 2 * /
6
a+b*c-d
(p-q)/q/d
a+b*c=d-e
t+t+t+t+t
s=s=s=s=s
(((((z)))))
3
R 3 = < >
L 3 & | ^
L 4 + - * /
1
a>b*c=d<e|f+g^h&i<j>k
```

Output for the Sample Input

```
((a+(b*c))-d)
(((p-q)/q)/d)
((a+(b*c))=(d-e))
((((t+t)+t)+t)+t)
(s=(s=(s=(s=s))))
z

(a>((b*c)=(d<((((e|(f+g))^h)&i)<(j>k)))))
```

Problemsetter: Yuta Kitamura

Problem K
電波基地

Time Limit: 3 sec

東京大学プログラミングコンテスト
31 May 2008

20XX 年，長年の研究が実を結び，無線によるエネルギー送受信の技術が実用化された．この技術を利用することで，コストが見合わず今まで電力線を引くことができなかった過疎地にも電力を供給することが可能となった．この技術には少しくせがあって，どこまでも遠くにエネルギーを送ることができるのだが，特定の位置でしか電波の受信ができないという制約が存在する．つまり，世界を 2 次元平面で表し，y 軸正の方向を北，x 軸正の方向を東とすると，x 座標・y 座標がともに整数の地点でしか電波を受信することができない（位相の問題である）．また，基地（電波の送受信が可能な設備）から見て 8 方位（東・西・南・北・北東・北西・南東・南西）の方向にしか電波を送信することができない．

この制約のために，目標の場所に電波を直接供給できない場合があるが，これは中継基地を置くことで解決可能である．例えば，座標 (0, 0) から (3, 7) へは直接エネルギーを送ることができないが，(3, 3) に中継基地を置くことで解決できる．

あなたは電力会社のエンジニアで，顧客からある場所に基地を用意してほしいと言われている．ただし，ある箇所が故障しただけで電力が止まってしまうという問題を避けるため，新たに基地を敷設する際には，既に建っている 2 箇所以上の基地からエネルギーを受け取れなくてはいけない．また，基地は同時には 1 つしか建てることができず，同じ場所に基地を 2 つ置くことはできない．既存の基地の位置が 2 つ与えられるので，顧客の要望を満たすためには最低でもあと何件の基地を建てる必要があるかを求めて欲しい．

Input

入力は複数のデータセットからなる．

入力の最初の行にデータセット数 N ($0 < N \leq 300$) が与えられる．以降の N 行に，それぞれのデータセットが与えられる．

1 つのデータセットは 1 行の文字列からなり，以下のようにスペースで区切られた 6 つの整数が与えられる．

$$x_1 \ y_1 \ x_2 \ y_2 \ X \ Y$$

(x_1, y_1) と (x_2, y_2) は既存の 2 つの基地の座標であり，(X, Y) は目的地の座標である．入力で与えられる x, y 座標値は $-100000000 \leq x, y \leq 100000000$ を満たす．既存の 2 つの基地の座標は異なっていることが保証されている．

Output

データセットごとに，敷設する必要のある基地の数を 1 行で出力せよ．

Sample Input

```
4
0 1 3 2 0 1
1 1 2 2 9 9
0 0 1 4 5 5
0 0 1 4 5 10
```

Output for the Sample Input

```
0
1
2
3
```

Problemsetter: Yusuke Konishi

Problem L
チクチクバンバン

Time Limit: 5 sec

東京大学プログラミングコンテスト
31 May 2008

屋根裏を整理していると，子供の時によく遊んだおもちゃが見つかった．このおもちゃはチクチクバンバンという名前で，サボテンを載せた貨物列車をレールから脱線しないように駅から駅まで運ぶというゲームだ．近所の子と一緒に熱中した懐かしい思い出が蘇り，掃除の途中であるのだが久しぶりに 1 回だけ遊ぶことにした．

このおもちゃの概観を図 7 に示す．

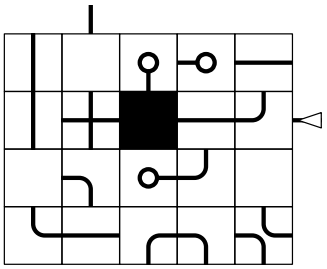


図 7　チクチクバンバンの概観図．

このおもちゃは，1 枚のフレーム，$W \times H - 1$ 枚のタイル，1 台の列車でできている．フレームには横幅 W，縦幅 H の長方形の窪みが空いていて，その窪みと隣接するように駅が 2 つ描かれている．タイルは 1×1 の大きさで上面にレールが彫ってあり，フレームの窪みにはめるとちょうど 1×1 の大きさの隙間が残るようになっている．この隙間と上下左右に隣接するタイルをスライドさせ，タイルと隙間の位置を入れ替えることによって，タイルの配置を変更することができるようになっている．

ゲームスタート時には，片方の駅に列車が置かれている（図 7 では列車を三角の印で表現している）．ゲームスタートと同時に列車は駅を出発し，レールに沿って移動し続ける．タイルの上のレールの形に関わらず，列車はあるタイルに入ってから出てくるまでにちょうど時間 1 を費やす．プレイヤーはタイルを適切なタイミングでスライドさせ，列車を脱線させないようにしなくてはいけない．脱線とは，レールが繋がっていない方向からタイルへ進入したり，タイルとタイルの隙間に落ちたり，駅以外のフレームに乗り上げたりすることである．列車がスタートした方とは異なる駅に到着することができたらゲームクリアである．

私はこのゲームをさんざんプレイしたことがあるため，タイルはいつでも好きな時に滑らかに，列車の速度に対して無視できる時間で動かすことができる．また，列車が乗っているタイルを列車ごとスライドさせるという高等テクニックも身につけている．そのため，慣れた手つきで格好良く軽々とクリアできるはずであった．ところが，長い間蒸し暑い屋根裏部屋に置いていたせいか，プラスチックが溶けてフレームとくっついてしまい，いくつかのタイルが動かなくなっていることがわかった．そのため，ゲームが非常に難しく，そしてより魅力的になってしまっていた．何度もプレイした結果，タイルの配置によってはクリアできるかどうかすら簡単にわからないこともあるということがわかった．

ここであなたにお願いがある．ゲームの初期盤面を与えると，その盤面がクリア可能かどうかを判定し，クリア可能なら列車の最短移動時間を求めるプログラムを書いて欲しい．もしプログラムが完成したなら，きっと私はこのゲームから足を洗うことができ，掃除の続きに戻ることができるだろうから．

Input

入力は複数のデータセットからなり，1 行目にデータセット数が与えられる．以降に，それぞれのデータセットが続く．

それぞれのデータセットは以下のようなフォーマットで与えられる．

W H
P_0 T_0 P_1 T_1
L_0
L_1
...
L_{H-1}

ただし，記号の意味は以下の通りである：

- W, H はそれぞれ箱の横幅・縦幅を表し， $1 \leq W, H \leq 20$ を満たす．
- (P_0, T_0) と (P_1, T_1) はそれぞれ異なる 2 箇所の駅の位置を表す．P は 0, 1, 2, 3 のいずれかの値をもち，それぞれ駅がフレームの上辺・右辺・下辺・左辺に存在していることを意味する．T は P の値によってそれぞれ次のような意味を持つ：
  - $P = 0, 2$ のとき，左からいくつの位置に駅があるかを表し，$0 \leq T \leq W - 1$ を満たす．
  - $P = 1, 3$ のとき，上からいくつの位置に駅があるかを表し，$0 \leq T \leq H - 1$ を満たす．
- $L_0, \ldots, L_{H-1}$ はそれぞれ長さ W の文字列であり，各文字は 1 つのタイルを意味し，全体で初期盤面の配置がどのようになっているかを表している．タイルにはそれぞれ図 8 のように名前がつけられており，動くタイルは名前が大文字で，動かないタイルは名前が小文字で与えられる．ここで，タイル D は，上下左右から列車が進入でき，進入してきた列車が直進してそのまま抜けていくタイルであり，また，E, F, G, H のタイルは，ある一方から来た列車が輪を描くように動き，同じ方向から出て行くというタイルである．初めの隙間の位置は “`#`” で与えられる．

Output

データセットごとに，列車が駅間を移動するのに最低限必要となる時間を 1 行に 1 つ出力せよ．もし駅間の移動が不可能なら “`-1`” を出力せよ．

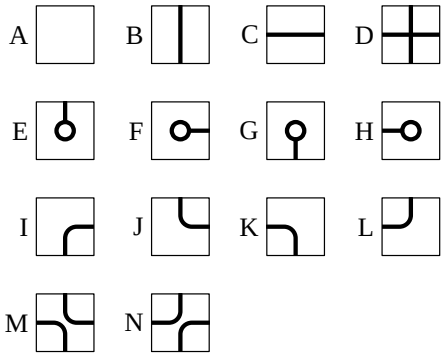


図 8　タイルの種類.

Sample Input

```
2
4 3
1 2 3 0
#bJG
KBBB
EeEA
6 5
3 4 1 2
aaaIMM
aaaIaA
N#BJDA
DaBaaa
MNBaaa
```

Output for the Sample Input

```
6
-1
```

サンプル入力の最初のデータセットの解決法を以下に示す．以下の説明において， t は列車がタイルに乗った瞬間からの時間を表す．図 9 において，列車は白い三角形の印で表し，固着したタイルはグレーで表している．

1. 左側にある駅からスタートする（図 9 (*a*)）.
2. $t = 0$: タイルを上・左と移動させる（図 9 (*b*)）.
3. $t = 0 – 1.5$: 列車を出発させ，時間が 1.5 経過する（図 9 (*c*)）.
4. $t = 1.5 – 2.5$: タイルを右に移動し，時間が 1 経過する（図 9 (*d*)）.
5. $t = 2.5 – 3.5$: タイルを左・左・下・左・上と移動し，時間が 1 経過する（図 9 (*e*)）.
6. $t = 3.5 – 4.5$: タイルを右・右と移動し，時間が 1 経過する（図 9 (*f*)）.
7. $t = 4.5 – 5.5$: タイルを左・左と移動し，時間が 1 経過する（図 9 (*g*)）.
8. $t = 5.5 – 6$: タイルを右・上・左・下と移動し，時間が 0.5 経過する（図 9 (*h*)）.

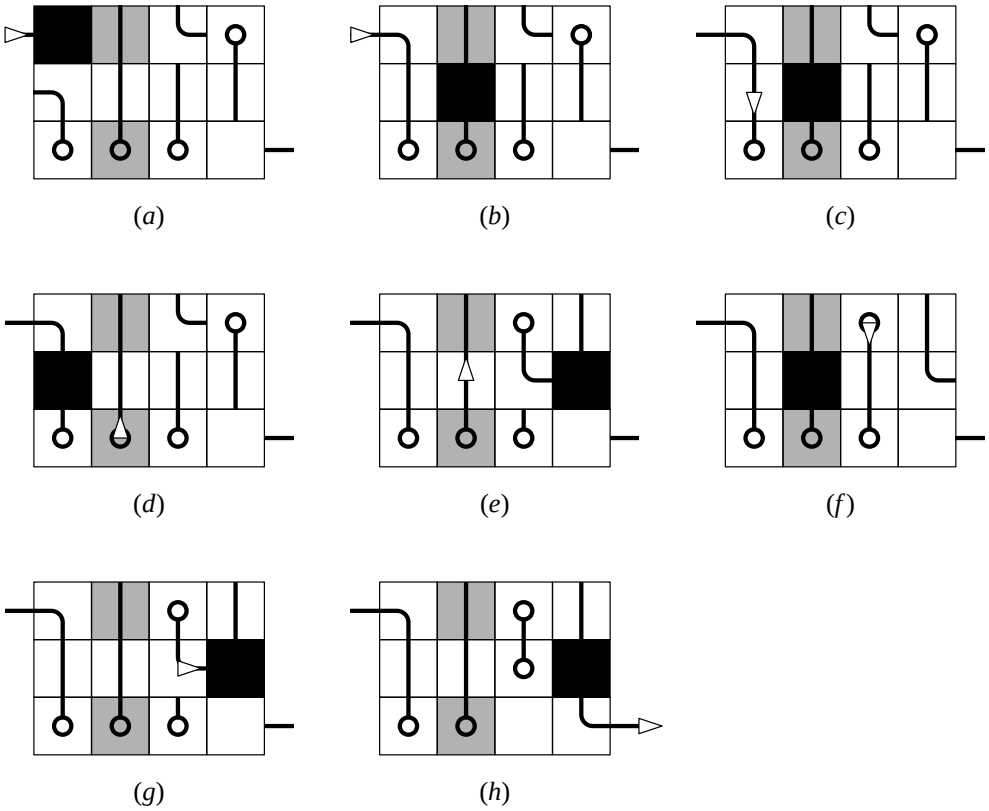


図 9　サンプル入力の最初のデータセットの解決法.

Problemsetter: Yusuke Konishi