2000 Denison Spring Programming Contest
Problem 1
Chutes and Ladders

The childrens' game *Chutes and Ladders* is played by moving pieces on a 10-by-10 board according to the roll of a pair of dice. If a player lands on a square that is the bottom of the ladder, she can move up to the square at the top of the ladder. If she lands on a square that is the top of a chute (a slide), the player moves to the square at the bottom of the chute. Each square is exactly one of the following: the top of a ladder or chute, the bottom of a ladder or chute, or a neither. The player starts on square 1 (which will not be the bottom of a ladder) and tries to be the first to land exactly on square 100 (which will not be the top of a chute). Recall that when rolling a pair of dice, the possible outcomes are from 2 through 12. When rolling a number, the player advances exactly that number of squares, then follows any ladder or chute from that landed upon square. If a roll would move the player past square 100, the roll is ignored and the player loses her turn.

Here, each square on the board will be given an integer. If the number is 0, then this square is either a normal square, the top of a ladder or the bottom of a chute and so nothing special happens if a player lands on this square. If the integer is $n > 0$, the square is the bottom of a ladder and if the player should land on this square, she would advance n squares to the top of the ladder. If the integer is $n < 0$, the square is the top of a chute and if the player lands on this square, she would go back n squares. In the boards you will have, you are guaranteed no ladder or chute will cause the player to move off the board. Furthermore, you are guaranteed that it is possible to win the game.

The board will be given as a series of 100 integers, 10 per row, indicating squares 1 through 100 in that order. For a given board, you are to determine the minimum number of rolls needed to win the game. **Warning:** Do not try to generate all possible rolls as this will exhaust your allotted running time.

There will be multiple boards inputted for this problem. The first line of input will be a positive integer k. There follows k boards, given in the format described above. For each board you should generate one line of output of the form:

`It will take at least` r `rolls to win this game.`

Of course r should be replaced by the appropriate number.

EXAMPLE:

Input:

```
1
0 0 42 0 0 0 57 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 -21 0 0 0 0 0 0 0
0 0 50 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 -19 0 41 -41
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 4 0 0 0 0 0 0
```

should generate output:

```
It will take at least 3 rolls to win this game.
```

2000 Denison Spring Programming Contest
Problem 2
Rabbit

In the poker game Rabbit, every player is dealt 5 cards and then 6 community cards are dealt face down in the center of the table in a 2-by-3 array (called the *tableau*) and are revealed one at a time. In the low version of this game (Low Rabbit), each player makes their lowest 5-card poker hand by playing exactly 3 cards from their hand with any of the 2-card columns from the tableau or playing exactly 2 cards from their hand with any of the 3-card rows from the tableau. The rank of a low 5-card poker hand is determined by the value of the largest card in the hand, providing the hand contains no pairs, is not a straight and is not a flush. (A pair is two cards of the same rank (such as two tens). A straight is 5 cards in consecutive order (such as 3-4-5-6-7). Note that an ace counts as a 1, in low poker. A flush is 5 cards all of the same suit.) Thus the lowest possible hand is ace-2-3-4-6, not all of the same suit. We'll call this hand *the very best hand*.

In order to win Low Rabbit, you must have a *qualifying hand*, which is a low hand whose rank is no higher than an 8. (Of course, this hand may not be a straight nor a flush.) The problem here is given a player's hand and the tableau, to determine the player's lowest possible qualifying hand, if one exists.

For example, if a player hand is 3H, 2S, JC, 2D, 7H and the tableau is
4H 9H 3D
AH 5H 7C

then the player's best hand is made by taking the 3, 7 and either 2 from his hand with the first column, giving a qualifying hand of rank 7. A player with a hand of 2H, 5H, 7H, KC, KD for the same tableau would not have a qualifying hand.

Input to this hand will be a series of n hands with tableaus. The first line of input will consist of the integer n. There will follow $3n$ lines of input, each set of three lines will represent a problem instance. The first line will be the player's hand, in the form

V_1S_1 V_2S_2 V_3S_3 V_4S_4 V_5S_5

each pair V_iS_i represents one card and the pairs are separated by a single blank. V_i represents the value of the card (either A, 2, 3, 4, 5, 6, 7, 8, 9, T, J, Q or K) and S_i represents the suit of the card (either H, C, S or D). The second line will give the first row of the tableau, left to right. The third line will give the second row of the tableau, left to right.

Each problem instance should generate one line of output. The output should be one of the following three lines (left-justified):

```
No qualifying hand.
The player has the very best hand.
The player has a qualifying hand of rank r.
```

In this last output line, r should be replaced by the appropriate rank.

EXAMPLE:

Input:

```
3
4H 9D 3H AH AS
8S 4D 5S
JH KH AD
4H 9D 2H AH AS
3D 4D 5S
JH KH AD
4H 9D 3H AH AS
2D 4D 5S
6H KH AD
```

should generate output:

```
The player has a qualifying hand of rank 8.
No qualifying hand.
The player has the very best hand.
```

2000 Denison Spring Programming Contest
Problem 3
Hospital Beds

A hospital is trying to minimize the number of beds needed in its emergency room (ER). A patient arrives in the ER, is treated and then either released or transferred to another ward. The treatment time is usually not very long. The hospital has records for the ER for the past few years. You are to write a program that processes the records for a number of days and prints out summary information for each day. Each record consists of the arrival time and length of stay in the ER for one patient. These two values are each in the format `hh:mm`, where `h` and `m` are digits. Thus the record `17:10 02:30` indicates a patient arriving at five-ten p.m. and staying for two hours and thirty minutes.

Arrival times and length of stay times are both recorded to the nearest 10-minute mark. At the end of a stay, it takes an additional 10 minutes for the bed and area to be prepared for the next patient. To simplify things, we'll assume that the ER is empty at the beginning of the day, but it may not be empty at the end of the day (that is, at time `24:00`). (However, there will be no "carry over" patients to the next day; each new day starts out with empty beds.) The summary information you are to produce is the maximum number of beds needed during the day.

Input for this program will be the positive integer n on a single line. There then follows information for n days. The information for one day consists of the integer m ($m \geq 0$) on one line, followed by m lines, each containing the arrival time and staying time of one patient, separated by one blank and in the format described above. The records for one day will be sorted by arrival time. Each day should generate one line of output of the form:

`The maximum number of beds used on this day was` b.

Of course b is replaced by the appropriate number.

EXAMPLE:

Input:

```
1
9
00:10 03:00
00:10 12:00
03:40 04:20
08:00 20:00
13:20 05:10
17:10 06:00
20:00 05:20
23:00 00:20
23:10 00:10
```

should generate output:

```
The maximum number of beds used on this day was 5.
```

2000 Denison Spring Programming Contest
Problem 4
D–

The programming language D– (successor to C++) uses 3 different pairs of symbols to delimit blocks of code: { and }, (and), and [and]. We'll call all of these *parens*. The blocks of code may be nested and each left parens — {, (or [— must be matched with the right parens of the same type. If a left parens is matched up with a right parens of a different type then there is a *type mismatch*. The rest of D– is of no concern except the very last line of the program is `THEEND`, left-justified on the line. All lines following this are ignored.

You are to write a preprocessor for D– that checks only that all blocks of code are delimited properly and report any errors. There will be three types of errors: 1) a type mismatch, 2) a right parens that has no matching left parens, and 3) a left parens with no matching right parens.

When you discover a type mismatch (a type 1 error) you should print a message of the form:

```
The left parens [ on line 5 is mismatched with the parens ) on line 23.
```

Of course the actual parens and line numbers should given appropriate values. If you discover a type mismatch, print the error message and then continue processing the program as if the parens did indeed match.

When you discover a type 2 error you should print out a message of the form:

```
The right parens ) on line 27 did not have a matching left parens.
```

Again, the actual parens and line number should be given appropriate values. When you detect a type 2 error, print out the error message and proceed as if the offending right parens did not exist.

When you discover a type 3 error you should print a message of the form:

```
The left parens [ on line 8 did not have a matching right parens.
```

Again, the actual parens and line number should be given appropriate values. Note that you will not be able to detect this error until you reach the end of the program.

Assume the first line of the program is line number 1 and that the program is indeed properly terminated with the `THEEND` statement. Furthermore, assume that the program contains no comments. Input for this program will be one candidate D– program of no more than 100 lines, with no more than 80 characters per line, and no more than 500 parens. The program will contain no blank lines. Output should be the error messages in any order.

EXAMPLE:

Input:

```
Program Example;
int x;
{ Some commands go here
   if ( x > 0 then
    [ input x;
      output x+1;
    }
THEEND
```

should generate output:

```
The left parens [ on line 5 is mismatched with the parens } on line 7.
The left parens ( on line 4 did not have a matching right parens.
The left parens { on line 3 did not have a matching right parens.
```

2000 Denison Spring Programming Contest
Problem 5
Ranking the Rankings

It's preseason and the local newspaper (the *Granville Tattler*) wants to publish a preseason ranking of the teams in the local amateur basketball league. When Scoop McGee, sports editor of the *Tattler*, gets the rankings from the selected local experts down at the hardware store, he's dismayed to find that there doesn't appear to be total agreement and so he wants to publish a composite ranking.

He's decided to use the method of assigning values to each place in the rankings. For example, if there are 5 teams, he will give 5 points to each team ranked first, 4 to each second, and so on, then add up the scores for each team. The one with the highest number of votes is ranked first, the next highest second, and so on. There may be ties, of course.

For example, suppose there are only 3 teams and 5 voters that have given the rankings: `ABC`,`ACB`, `BAC`, `BCA` and `CBA`. Then A would get 10 points, B would get 11 points and C would get 9. Thus `BAC` would be the composite ranking.

Input for this program will be multiple problem instances. The 1st line will be the positive integer k. There follows k problems instances. Each problem instance starts with two positive integers n ($n \leq 26$) and m on a line, separated by one blank. This indicates that there are n teams and m voters. This initial line is followed by m lines, each containing a permutation of the first n upper case letters, left-justified with no spaces. For each problem instance your program should produce n lines of output, giving the composite ranking, followed by a single blank line. These lines should have the same format as the following example:

```
1.  B
2.  C
2.  D
4.  A
5.  E
```

In this example $n = 5$ and there was apparently a tie for second place. If there is a tie, you may list those tied in any order.

EXAMPLE:

Input:

```
1
5 4
ABDCE
BACDE
CABED
ACEBD
```

should generate output:

```
1.  A
2.  B
2.  C
4.  D
4.  E
```