
# Monash Collegiate Programming Contest

March 28, 2015

---

# Contest Problems

---

A: Assigning Seats
B: Mo's Path
C: Blue Chips
D: Daredevil Dave
E: Pressure Drill
F: Horse Racing

This Page Is Intentionally (Almost) Blank.

# A: Assigning Seats

The Airline for Cool Mathematicians (ACM) is a new and upcoming international airline. They pride themselves on many things, but none more than their seating policy: "If you are not seated with the other members of your family, everyone in your family will receive a full refund!" A family is considered to be sitting together if they are in contiguous seats in the same row of the aircraft. For simplicity of this problem, you may assume that there is no aisle on the aircraft.

Given the configuration of the aircraft, can you determine if it is possible for the ACM to assign the seats in such a way that they do not have to give a refund? You may assume that all families will have size at most four and that each row has at most four seats.

## Input

The first line of the input contains a positive integer $T$ ($T \leq 1000$), denoting the number of test cases to follow. Each test case will take place over two lines.

The first line will contain 4 integers describing the aircraft. The first integer on the line denotes the number of rows which have only 1 seat, the second integer on the line denotes the number of rows which have 2 seats, etc.

The second line will also contain 4 integers describing the sizes of the families who have purchased tickets. The first integer denotes the number of families of size 1, the second integer denotes the number of families of size 2, etc.

All numbers in the input will be nonnegative and less than 1000. You may assume that the number of passengers on the flight is less than or equal to the number of seats.

## Output

For each test case, output one line containing either "`Refunds`" or "`No Refunds!`" without the quotes.

## Sample Input

```
3
1 1 1 1
1 1 1 1
0 0 2 1
0 5 0 0
0 0 1 1
0 2 1 0
```

## Sample Output

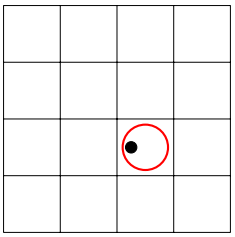
```
No Refunds!
Refunds
No Refunds!
```

This Page Is Intentionally (Almost) Blank.

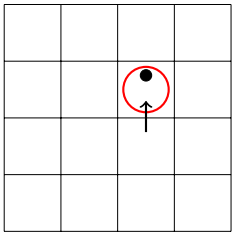
# B: Mo's Path

I have a robot vacuum at home. His name is Mo.[1] Watching him zoom around our carpets all day is rather amusing. Sometimes he will run into walls unexpectedly and sometimes he will miss large patches of the carpet. One day, as Mo was finishing his vacuuming, I wondered if he had actually gotten all of the carpet. Can you help me determine if he actually got every last bit of the carpet by tracing Mo's path?

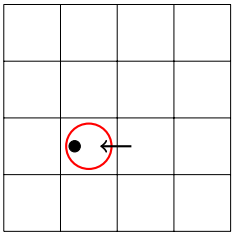
The room is $m$ metres by $n$ metres and you may think of the room as a grid where Mo will always be located in the centre of one of the squares. Mo is a circular vacuum and has a black spot on one side of the circle to denote which direction he is facing. For example, in the following, Mo is facing west.



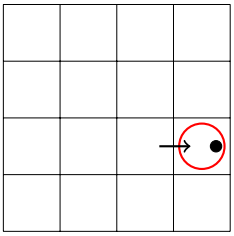
From this position, Mo may do one of four moves: Forward, Left, Right or Backward. In each of the four operations, Mo will turn his body (if necessary) and then move to one of the four adjacent squares. The new look of Mo after each of the four operations is shown below:



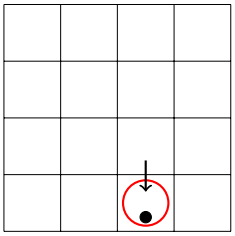
Right



Forward

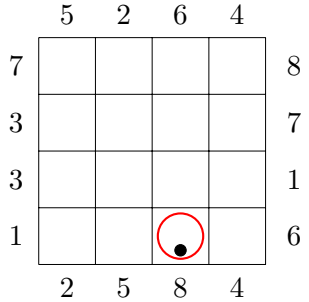


Backward



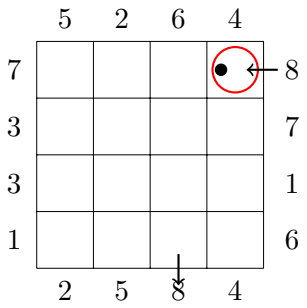
Left

Oh, and I forgot to mention that my room has many teleporters all along the walls! On each metre of wall, there is a teleporter which will take Mo from his current position to another teleporter somewhere else in the room. There are exactly $2(m+n)$ teleporters along the walls of the room and each one has a corresponding *transmission frequency* between 1 and $m+n$. There are exactly two teleporters with each valid transmission frequency. For example, here is a valid room:




[1] Named after the loveable character in the movie *WALL-E*.

When Mo enters a teleporter with a transmission frequency of $x$, he immediately appears in front of the other teleporter with the same transmission frequency (recall that there are exactly two teleporters with each transmission frequency, so his destination is unique). Mo will always be pointing *inwards* when he exits the teleporter. So in the example room above, if Mo were to go Forward, then he would go through the teleporter with a transmission frequency of 8 and come out the other teleporter with the same frequency. So his new location would be as follows:




## Input

The first line of the input contains a positive integer $T$ ($T \leq 100$), denoting the number of test cases to follow. Each test case will start with two integers $m$ and $n$ ($1 \leq m, n \leq 50$), denoting the number of rows and columns, respectively.

The following line gives the starting location of Mo. The line will start with two integers $a$ ($1 \leq a \leq m$) and $b$ ($1 \leq b \leq n$) giving the row and column of where Mo starts. Note that $(1, 1)$ is the top-left corner. Then, one of the words `North`, `South`, `East` or `West` will appear to to show which way Mo is facing originally.

The next line will contain $2(m+n)$ integers denoting the transmission frequencies of the sections of the wall starting at the top-left corner and working clockwise around. The transmission frequencies in the first sample input correspond to the example above. Each of the transmission frequencies will be between 1 and $m + n$ and there will be exactly two of each number.

The next line will contain the instructions that Mo must follow. The line will start with a nonnegative integer $k$ ($k \leq 10000$) denoting the number of instructions. Then the $k$ instructions will appear on that line. Each instruction will be one of `Forward`, `Backward`, `Left` or `Right`.

## Output

For each test case, output Mo's path throughout the room. You should output a grid of $m \times n$ characters with the following rules:

- The grid starts as an $m \times n$ array of dots (`.`).
- On the square that Mo starts, put an `S` and on the square that Mo finishes, put an `E`. If these two squares are the same, put an `X` instead.
- For all other squares that Mo travels through, you will place *movement markers*, which are one of the following three symbols: `|`, `-` or `+`. You should use the `|` if Mo entered the cell from either the north or the south and left out the opposite wall. Similarly, you should use the `-` if Mo entered the cell from either the east or the west and left out the opposite wall. The `+` should be used in all other situations to denote turning. We are interested in the last time that Mo passes over each location, so if Mo passes over a square that already has a movement marker, then it should be overwritten (see the second sample input).

Between each pair of test cases, output a blank line.

## Sample Input

```
2
4 4
4 3 South
5 2 6 4 8 7 1 6 4 8 5 2 1 3 3 7
1 Forward
5 5
4 2 North
1 2 3 4 5 2 3 4 5 1 6 7 8 9 10 7 8 9 10 6
8 Right Left Forward Forward Forward Forward Forward Forward
```

## Sample Output

```
...E
....
....
..S.

..|..
.E---
..|..
.S+..
.....
```

This Page Is Intentionally (Almost) Blank.

# C: Blue Chips

$N$ people are playing the following game: Everyone stands in a circle at equal distances from their 2 neighbours and starts with a certain number of blue chips. The game is played in $K$ rounds: In each round, you add up the number of chips that every person within a certain distance ($D$) from you has, and that is the number of chips that you start the next round with. Note that you do not consider the number of chips you currently have in this computation.

After the $K$ rounds, each player will get a score based on the number of blue chips they have. Player $i$'s score is the number of blue chips that Player $i$ has mod $N$. For example, if you have 16 blue chips, and $N = 7$, then your score would be 2. The winner is the player (or players) with the smallest score.

Note that when you are computing the distance between you and another person, you simply need to count the number of people between you and them. This means that your two neighbours are at distance 1 from you, the people beside them are at distance 2 from you, etc. You will always take the shortest way around the circle when computing the distance.

## Input

The first line of the input contains an integer $T$ ($1 \leq T \leq 10$), the number of test cases. Then follow $T$ test cases, each on two lines. The first line contains 3 integers: $N$ ($2 \leq N \leq 50$), $K$ ($1 \leq K \leq 10^9$) and $D$ ($1 \leq 2D \leq N$). The second line contains $N$ integers $X_i$ ($1 \leq X_i \leq 1000$), the starting number of blue chips each player is holding, in clockwise order around the circle.

## Output

For each test case, output two lines: The first line should contain the score of the winner(s). The second line contains a list of all winners separated by a space in increasing order. The first person listed in the input is player 1, the second person in the input is player 2, and so on. The last player in the input is player $N$. Player 1 and player $N$ are neighbours.

## Sample Input

```
1
3 3 1
1 2 2
```

## Sample Output

```
1
2 3
```

This Page Is Intentionally (Almost) Blank.


# D: Daredevil Dave

Daredevil Dave has been locked in a room at the top of a tower by the evil Oswald Hunter-Smith! Oswald plans to keep Dave there forever, and prevent him from having fun. The tower has only one window, much too high above the ground for even Daredevil Dave to survive a fall. However, the room is full of short lengths of elastic, left over from when the tower was used as an underpants factory. Dave decides to tie together a number of lengths of elastic to form a bungee cord to enable him to escape. He must be careful, though - too long and he will smash into the ground, too short and he will be left dangling forever. Every knot that Dave ties will use up some of his elastic. Can Dave safely bungee out of the window?

Each piece of elastic can only be used once per bungee cord. Lengths may appear multiple times, in which case they can each be used to form the bungee cord. Dave can only safely bungee out of the window if the sum of the lengths of elastic, less the elastic used in knots, exactly equals the height of the tower. If the cord is even 1 m too short, Dave will be left hanging above the ground, and if it is even 1 m too long, he will sprain an ankle on landing and be unable to run away.

A knot is required to attach two pieces of elastic to each other. No knot is required to attach the top piece to the window frame, or to attach the last piece to Dave's belt. Any length of elastic shorter than the knot length cannot be used to tie a knot.

## Input

The first line of input contains $n$ ($1 \leq n \leq 50$), the number of test cases. Each of the next $n$ lines represents one test case. Each test case line starts with $h$ ($1 \leq h < 10^5$), the height of the tower in metres; $v$ ($0 \leq v < 100$), the length of elastic used to tie a knot; $k$ ($1 \leq k < 1000$), the number of pieces of elastic Dave can find in the tower room; followed by $k$ integers representing the lengths of each of the $k$ pieces of elastic when stretched. Each length will be no more than 100.

## Output

For each test case, output `yes` if Dave can tie together some subset of the elastic pieces to form a bungee cord exactly the height of the tower, and `no` otherwise. Each line of output should be terminated by a newline character. Note that one piece of elastic does not require any knots, two pieces of elastic require one knot, three pieces require two knots, etc.

## Sample Input

```
4
6 1 4 2 2 2 2
6 1 5 2 2 2 2 2
12 0 4 1 7 4 3
16 0 4 7 4 3 1
```

## Sample Output

```
no
yes
yes
no
```


# E: Pressure Drill

Digging a deep hole has always been a problem. An average person can dig 1 metre every hour with a shovel. So if you wanted to dig a hole that is 50 metres deep, it would take you 50 hours. Thankfully, there is a solution: a pressure drill. The pressure drill works in the following way: It is dropped from the top of the hole and gains enough speed to exactly double the depth of the hole.

I really need to dig a hole in my back yard, and I need to do it quickly. Can you tell me the quickest way that I can dig a hole of size exactly $k$ using only a shovel and the pressure drill? You may assuming that each usage of the pressure drill also takes one hour.

## Input

The first line of the input contains a positive integer $T$ ($T \leq 100$) denoting the number of test cases to follow. Each test case is a single integer $k$ ($1 \leq k \leq 10^9$) denoting the depth of the hole that I wish to dig.

## Output

For each case, output a series of instructions for me on one line which will result in a hole of depth $k$ and be the quickest way possible to do it. Each instruction must be one of `dig` or `drill`. When you use `dig`, the depth of the hole increases by exactly one and when you use the `drill`, the depth of the hole exactly doubles. You will always start off with no hole (a depth of 0). Separate each instruction by a single space. If there are multiple correct solutions which give the depth of $k$, you may output any of them.

## Sample Input

```
3
1
2
3
```

## Sample Output

```
dig
dig drill
dig dig dig
```

This Page Is Intentionally (Almost) Blank.


# F: Horse Racing

Darcy and Joshua each have 10 horses and they love to race against each other. They usually have 10 rounds of racing, and they agreed that no horse can appear in more than one round. Which means that the outcomes of their racing events only depend on the order that the horses are put into the competition.

After several racing events, Joshua observed that Darcy always uses better horses earlier. For example, let $A, B, C$ be 3 of Darcy's horses, and assume A is faster than B, and B is faster than C, then Darcy will let $A$ compete in the first round, let $B$ compete in the second round, and let $C$ compete in the last round.

Joshua wants to win as many rounds as possible and he wants to know the maximum number of rounds he can win. Can you help him figure this out?

## Input

The first line of the input gives the number of test cases, $T$ ($1 \leq T \leq 1000$). $T$ test cases follow. Each test case has two lines. The first line provides speed of Darcy's horses, 10 integers separated by a single space. The second line provides speed of Joshua's horses, 10 integers separated by a single space. The speeds of horses are all positive integers no greater than 1000.

## Output

For each test case, output one line containing the maximum number of rounds Joshua can win if he plays optimally.

## Sample Input

```
4
20 40 60 80 10 30 50 70 90 100
1 2 3 4 5 6 7 8 9 10
20 40 60 80 10 30 50 70 90 100
71 81 91 101 11 21 31 41 51 61
10 30 50 70 90 100 20 40 60 80
11 17 21 35 39 42 67 87 100 90
10 10 10 10 10 10 10 10 10 10
5 6 7 8 9 10 11 12 13 14
```

## Sample Output

```
0
10
8
4
```

This Page Is Intentionally (Almost) Blank.