STATE UNIVERSITY
OF BANGLADESH
join the trendsetter

SUB INTER UNIVERSITY PROGRAMMING CONTEST 2016

February 26 & 27

ORGANIZED BY | SUB COMPUTING CLUB | Department of Computer Science & Engineering

This set contains 16 Pages and 11 Problems

Rules for SUB IUPC 2016

Solutions to problems submitted for judging are called runs. Each run is judged as accepted or rejected by the judge, and the team is notified of the results.

A contestant may submit a clarification request to judges only through the CodeMarshal clarification system. If the judges agree that an ambiguity or error exists, a clarification will be issued to all contestants. Judges may prefer not to answer a clarification at all in which case that particular clarification request will be marked as IGNORED in the CodeMarshal clarification page.

Contestants are not allowed to bring in any kind of electronic or storage devices with them. This includes any kind of cellphones, smart phones, pocket media devices, CD/DVD, flash drives, calculators, smart watches or any other kinds of electronic devices that can connect to internet or store data. Contestants can leave these to their coaches or submit them to volunteers before the contest starts. **Failure to adherence to this clause under any condition will lead to strict disciplinary retaliation and immediate disqualification.**

Contestants are not to converse with anyone except members of their team and personnel designated by the organizing committee while seated at the team desk. **They cannot even talk with their team members when they are walking around the contest floor to have food or any other purpose.** Systems support staff or judges may advise contestants on system-related problems such as explaining system error messages.

While the contest is scheduled for a particular time length (five hours), the contest director has the authority to alter the length of the contest in the event of unforeseen difficulties. Should the contest duration be altered, every attempt will be made to notify contestants in a timely and uniform manner.

A team may be disqualified by the Contest Director for any activity that jeopardizes the contest such as dislodging extension cords, unauthorized modification of contest materials, distracting behavior or communicating with other teams. The **judges on the contest floor** will report to the Judging Director about distracting behavior of any team. **The judges can also recommend penalizing a team with additional penalty minutes for their distracting behavior.**

Teams can bring up to **200 pages of printed materials** with them and they can also bring five additional books.

With the help of the volunteers, the contestants can have printouts of their codes for debugging purposes. **Passing of printed codes to other teams is strictly prohibited.**

The decision of the judges is final.

Teams should inform the volunteers/judges if they don't get verdict from the CodeMarshal within 5 minutes of submission. Teams should also notify the volunteers if they cannot log in into the CodeMarshal system. These sort of complains will not be entertained after the contest.

A Devil Number

1, 3 & 5 are called devil digits. A number is a devil number if it consists any devil digit.

You are given a positive integer number. You need to determine if the number is a devil number or not.

Input

First line contains an integer number **T (**$\mathrm{T} \leq 20$**)** denoting the number of test cases. Then **T** lines follow. Each line contains an integer **N (**$1 \leq \mathrm{N} \leq 100000$**).**

Output

For each case you have to print the case number. Then 'Yes' or 'No' without quotes. Print 'Yes' if the number is a devil number else print 'No'. See the samples for exact formatting.

Sample Input	Sample Output
4 1 7 1108 2884	Case 1: Yes Case 2: No Case 3: Yes Case 4: No

B Blast the tent

Today I'm talking about a young freedom fighter of Bangladesh named Altaf. He was not only a fighter but also a brilliant and sensible person. He put on a coat with **B** pockets to carry bomb with different capacity. He used to spend a lot of time to make a proper plan before throwing a single bomb at the enemies. Thus, by avoiding all risks of getting caught he maximized the loss of the enemy. He determined the number of enemies in all tents and targeted zero or more tents to throw bombs at them with appropriate capacity that maximized the number of enemies killed.

He figured out that,

- In the enemy camp, the soldiers settle down their tents with a safe distance from each other. The tents are divided in **R** rows and **C** columns.
- When a tent is attacked, other soldiers of the same and adjacent columns increase their security making impossible for Altaf to attack those tents.
- If some soldiers of the attacked tent remain alive Altaf will get caught.
- Tents might be attacked randomly considering all of the safety issues.

Now consider you are in his position. You need to blast some tents to kill maximum enemies without getting caught by them. I know you are so brilliant and have ability to determine his answer more optimally. What would you do if you were there with a liability to save your motherland?

Input:

First line of the input data will be **T** (**T** ≤ 60), which denotes the number of input sets. First line of a single input set will be the number of bombs **B** (1 ≤ **B** ≤ 10) available in your pocket. Following this line will have **B** space separated integers $\mathbf{B_i}$ (1 ≤ $\mathbf{B_i}$ ≤ 1000) denoting the capacity to kill number of enemies. Next line will have 2 space separated integers **R** and **C** (1 ≤ **R**, **C** ≤ 30) followed by R lines with C space separated integers which denotes the number of enemies in that tent. Number of enemies in any tent will not exceed 1000.

Output:

For each case, print the case number and maximum number of enemies you can kill throwing the bombs without being caught.

Sample Input	Sample Output
2 5 1 2 3 4 5 4 5 2 8 6 1 2 1 5 4 9 3 2 4 6 9 7 3 5 9 4 2 3 35 100 50 3 5 2 3 10 80 90 30 50 20 10 4 3 90 4 3 6	Case 1: 10 Case 2: 140

C Business

Saurav runs a business. Right now, he has got **N** pending task proposals. For any *i-th* ($1 \le i \le N$) task, he will be paid $\mathbf{P_i}$ which will be reduced by $\mathbf{R_i}$ for each day it takes to finish and he has calculated that $\mathbf{D_i}$ days will be required to do only that task.

For every *j* ($1 \le j \le N$), Saurav wants to know $\mathbf{E_j}$, maximum earnings from first *j* tasks. He cannot work on multiple tasks on the same day. Day count starts from first day for all tasks.

Input:
Input starts with a line containing an integer **T** (≤ 12), the number of test cases. Each test case begins with an integer **N** ($\le 10^5$) followed by N lines, one line per task from 1 to N. Each of these N lines contains three space separated positive integers $\mathbf{P_i}$ ($\le 10^8$), $\mathbf{R_i}$ (≤ 100) and $\mathbf{D_i}$ (≤ 100). Summation of all N among all test cases will not exceed 500,000.

Output:
Print T lines in the output, one for each test case. Begin each line with "Case x:" where x is the case number followed N space integers which are $E_1, E_2, \ldots, E_N$.

Sample Input	**Sample Output**
2 1 5 2 3 3 29 5 4 6 1 3 7 2 2	Case 1: -1 Case 2: 9 8 1

D Home Redecoration

After a lot of discussion about the old furniture in your house, your wife as finally convinced you to sell all of it and buy new ones. You've unwillingly accepted this proposal of your whole home redecoration but you are BROKE! I guess it's time to use your skills as a BLU, the master thief (which is your secret identity). Also you are very tired and irritated about your neighbors and want to teach them a lesson. What a good opportunity to achieve two goals in one swift strike.

You have made a list of items to steal from your neighbors (you will take each item exactly once), you also have list of what items each neighbors own that you need to redecorate your home. Each item can be available at multiple neighbors home and some may be costlier than the others.

You want to teach all of your neighbors a lesson but don't want them to be so much afraid that they left the neighborhood. You have made a mental note that you will steal from all the neighbors (take at least one item from each of them) but won't take more than X items, where X may vary from neighbor to neighbor.

After fulfilling these conditions, you also want to make your wife the happiest, so you want the items you steal to cost as much as possible.

Input

Input starts with number of test cases, **T** (**T** ≤ 30).
Each test case begins with two integers, **n** (0 < **n** ≤ 100) and **m** (0 < **m** ≤ 100). Where **n** denotes the number of items you have to steal and **m** denotes the number of neighbors you can steal from. Next line will contain m integers $\mathbf{X_i}$ ($1 \le \mathbf{X_i} \le 100$), each denoting the maximum number of items you can steal from the $\mathbf{i^{th}}$ neighbor.
Next like will contain an integer **r** ($1 \le \mathbf{r} \le 1000$). Each of the r lines contains 3 space separated integers **u** ($1 \le \mathbf{u} \le$ n), **v** ($1 \le \mathbf{v} \le$ m) and **w** ($1 \le \mathbf{w} \le$ 1,000). Which denotes that, the $\mathbf{u^{th}}$ item can be found at the $\mathbf{v^{th}}$ neighbor's house and costs **w** units.

Output

For each case, print the case number and the total cost of the furniture you steal or "Impossible" if you can't steal all the items in your list while fulfilling all the conditions.

Sample Input	Sample Output
2 2 2 2 2 2 1 2 5 2 1 5 2 2 2 1 2 1 1 10 1 2 5	Case 1: 10 Case 2: Impossible

E Acme Paints Bangladesh

Life is colorful. So is everything around you. Acme Paints Bangladesh company keep this into their mind. They build a mobile app where clients can choose and paint custom objects with their desired colors. For you blue is blue, but the equation is not that simple to everyone. To them it may be periwinkle or crayola blue. For this purpose, the company maintains a custom color strip from where users pick color or make some customize. As these clients are a bit much choosy (yes indeed a bit weird too) after customization they wants to know, number of different nonzero length contiguous colors subsequence possible from this color strip. You can assume the initial strip is empty. Then colors are added, deleted, replaced or shifted. Each color is represented by a lowercase English alphabet. In the customization process a user can do the following operations,

- **`I c x t`**: Insert **t** units of color **c** starting at position **x**.
- **`D x y`**: Remove colored units from segment position **[x-y]** inclusive
- **`R c x y`**: Replace colored units from segment position **[x-y]** inclusive with the new color **c**
- **`S t`**: Cyclic right shift the colored segment by **t** times (Each cyclic shift will move the strip 1 position to the right)

Let at any moment the colored strip is **aabc**. Then following commands are given:

- **`I d 2 3`**. So current strip becomes, **adddabc**
- **`D 3 5`**. So current strip becomes, **adbc**
- **`R b 2 3`**. So current strip becomes, **abbc**
- **`S 2`**. So current strip becomes, **bcab**

Input

Input starts with T, number of test cases ($T \leq 20$). Then each case follows. Each test case starts with C, number of commands ($1 \leq C \leq 1{,}00{,}000$) followed by each command, C, where C is a character which can be (`I`, D, R, S) and then corresponding values for that command. For each insert command number of strips to color **t** holds the condition ($1 \leq t \leq 10{,}000$), You can assume, all other given position values are valid for strip at that moment. For number of cyclic shifts t, ($1 \leq t <$ *length of current strip*). All the positions are 1 based and for any segment range x and y, $x \leq y$. After performing all commands the final length of the strip will not exceed 100000.

Output

For each case print the case number following by the number of distinct contiguous subsequence of nonzero length found on the final color strip.

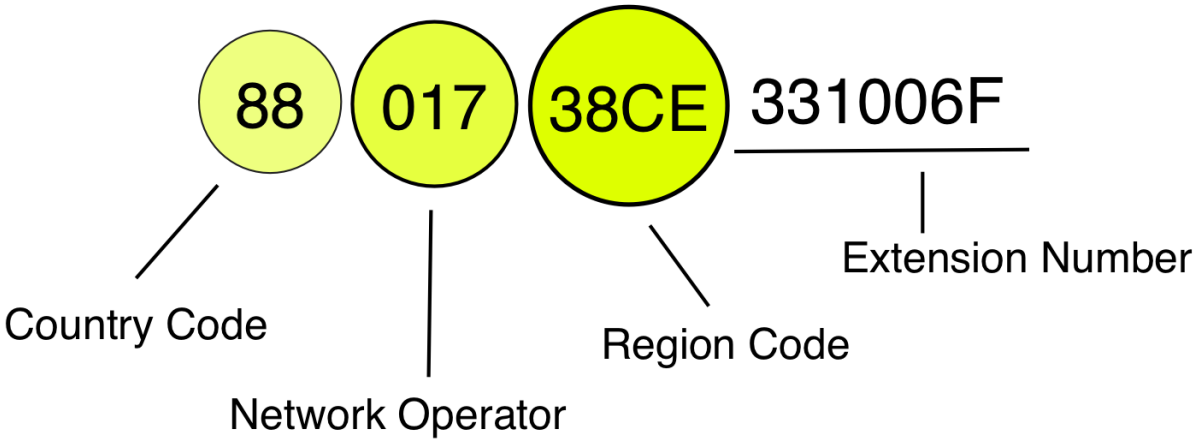
Sample Input	**Sample Output**
2 3 I a 1 7 R c 2 3 S 2 5 I a 1 4 I b 2 3 D 1 2 R c 2 3 S 3	Case 1: 21 Case 2: 13

Explanation for first case:

At query one we insert seven ‘a’ characters at position 1, so we get aaaaaaa. At second query we replace second to third place inclusively with character ‘c’. So we get, accaaaa. At third query we make two cyclic shifts at right direction and get aaaccaa. Number of different contiguous nonzero length subsequence for this strip is 21.

F Counting Phone Numbers

A valid phone number in Mars is as the following example



The properties of a valid phone number are:

1. Country Code is a string of length 2 having only hexadecimal digits [0-9, A-F]. It may be preceded with a single '+' sign. Some valid examples are +88, 42, +FF.
2. Network Operator Code is a string of length 3 having only hexadecimal digits [0-9, A-F]. In a phone number it may be encapsulated with only one pair of braces "()" or may be preceded with a single '-' sign. Some valid examples are 03A, (A10), -58D.
3. Property of Region Code is same as Network Operator code except its length is 4.
4. Property of Extension number is same as Network Operator code except its length is 7.

Your task is to read $\boldsymbol{n}$ strings and you'll have to check whether these strings are valid phone numbers. After discarding the invalid phone numbers you'll have to print for each country its country code, code of all the distinct network operators and code of all distinct regions and number of distinct phone numbers of that country. Country codes needs to be sorted lexicographically and for each country code both network operator codes and region codes has to be sorted in lexicographical order. Note that a phone number is invalid if any of the Country Code, Network Operator, Region Code or Extension Number is invalid.

Input:

Input starts with an integer $\boldsymbol{T} \leq \mathbf{100}$, denoting the number of test cases. Each case starts with a number $\boldsymbol{n}$ $(\mathbf{1} \leq \boldsymbol{n} \leq \mathbf{200})$ denoting the number of strings. The next $\boldsymbol{n}$ lines are the strings that has $\boldsymbol{C}$ characters where $\mathbf{1} \leq \boldsymbol{C} \leq \mathbf{32}$ and it includes all the uppercase letters of the alphabet and the numeric letters and any of the following characters

! , . : ; ? + ( ) -

Output:
For each test case, print the case number in a single line. Then for each country print the required information. For each test case you will get at least one valid phone number. The output pattern should follow the Sample Output below.

Sample Input	Sample Output
2 9 +10?1FC.-880A9B1D3DA +966D3230B8510E71 101FE-881A9B1C3DB 101FE-880D9B1D3DA +08(669)(92A5)(00E3AAB) +96-F3A(F699)-060D820 08(669)-92A5(C0E3A53) ++88(017)-38CE(331006F) 22(777)((38CE))(100331A) 4 +SUB-CONTEST(2016) +BD-ACM(2016)-IUPC +TRY+HARD-GET,SUCCESS 88-017-AAC1(0987456)	Case 1: Country Code: 08 Network Operator Codes: 669 Region Codes: 92A5 Total Phone Numbers: 2 Country Code: 10 Network Operator Codes: 1FE Region Codes: 880D 881A Total Phone Numbers: 2 Country Code: 96 Network Operator Codes: 6D3 F3A Region Codes: 230B F699 Total Phone Numbers: 2 Case 2: Country Code: 88 Network Operator Codes: 017 Region Codes: AAC1 Total Phone Numbers: 1

G Mirror Query

You are given n (n < 100,005) points in 2D plane, all are integer. You have to deal with m (m < 20,004) queries. They are of three kinds.

1. **r** *i j* x_0 y_0 x_1 y_1
   Send all the points from index i to j to their reflected points with respect to an infinite mirror that goes through x_0, y_0 and x_1, y_1 point.
   Constraints for the variables are: ($1 \le i, j \le n$), ($-100{,}005 < x_0, x_1, y_0, y_1 < 100{,}005$)

2. **t** *i j x y*
   Send all the points from index i to j to their translated points by adding (x, y) vector to each of them.
   Constraints for the variables are: ($1 \le i, j \le n$), ($-100{,}005 < x, y < 100{,}005$)

3. **q** *i*
   Output the coordinates (x, y) of the point at index i.

Input

Input starts with **T**, number of test cases (**T** ≤ 100).
Each case starts with **n**, number of points given (**n** < 100,005) following the coordinates (**x**, **y**) of each points where (-100,005 < **x, y** < 100,005).
Next line contains an integer **m**, number of queries (**m** < 20,004) followed by **m** queries. Each of the queries will be formatted as the three kinds of queries given above.

Output

Each of the case starts with the case number. Then for each '**q**' type query, print the required output. Error less than 10^{-2} will be ignored.

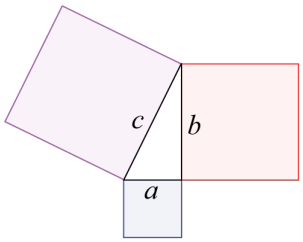
Sample Input	Sample Output
1 4 2 2 1 1 3 1 4 1 9 r 2 4 4 -1 10 -1 q 4 t 1 2 -1 3 q 1 t 1 2 1 -3 q 3 r 2 4 10 -1 4 -1 q 1 q 2	Case 1: 4.000000 -3.000000 1.000000 5.000000 3.000000 -3.000000 2.000000 2.000000 1.000000 1.000000

H Pythagoras Again?

The **Pythagorean theorem**, also known as **Pythagoras' theorem**, is a relation in Euclidean geometry among the three sides of a right triangle. It states that the square of the hypotenuse (the side opposite the right angle) is equal to the sum of the squares of the other two sides. The theorem can be written as an equation relating the lengths of the sides ***a***, ***b*** and ***c***, often called the "Pythagorean equation":

$$a^2 + b^2 = c^2$$

where ***c*** represents the length of the hypotenuse and ***a*** and ***b*** the lengths of the triangle's other two sides.



In this problem, you will be given the length of the hypotenuse ***c*** of a right triangle, find the other two sides ***(a, b)*** of the right triangle such that ***a, b are positive integers and (a ≤ b).***

Input

Input starts with an integer ***T (≤ 1000)***, denoting the number of test cases.

Each case will contain one positive integer ***c (1 ≤ c ≤ 500,000)***, the length of the hypotenuse of the right triangle.

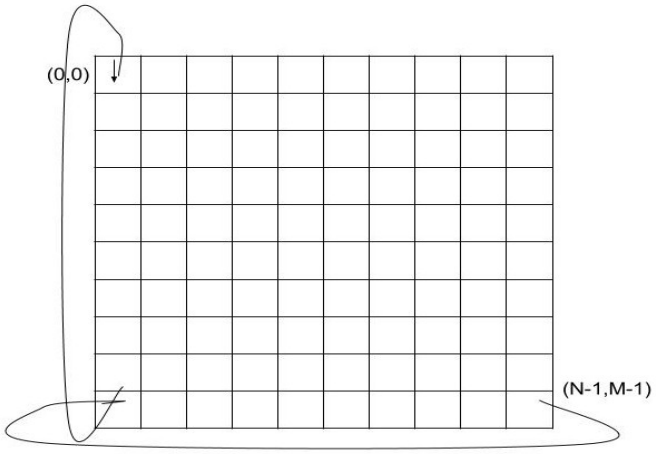
Output

For each case, print the case number first, then print the other two sides ***a*** and ***b***. If no such ***a, b*** can be found which satisfies the above condition print ***"Impossible"*** without quotes. If there are multiple pair of ***(a, b)*** exists, print the one with smallest possible ***a***.

Sample Input	Output for sample input
3 5 6 25	Case #1: 3 4 Case #2: Impossible Case #3: 7 24

I Snake Game

Farmer John created a snake game where in each move a snake can go to the cell forward or turn and go to the left or right cell. Sometimes a fruit appears and remains in the board for a limited amount of moves. The snake can eat the fruit in the board and grow in size by one unit. Each snake length unit is equal to the size of a cell on the board. Also the board is wrapped from end to end. Which means, if you go through the left side of the board, you will appear from the right side on the same row. If you go through the bottom side of the board, you will appear from the top side on the same column and vice versa. If on any move, the snake collides with itself, the game ends.



For any, N x M snake board the upper left-most cell is (0,0) and lower right-most cell is (N-1,M-1). The snake starts from (0,0) , initial length is one and facing downward direction.

There are two types of input. First one is movement, formatted as:

M D

where D is either F,L or R. Here F means go forward , L means turn left and go forward, R means turn right and go forward.

Another type of input is reveal of fruit, formatted as:

P x y t

Here, a fruit is revealed at (x, y) position which will be available for at most t moves. There will not be any input where two fruits will be available at the same time.

Input:

Input starts with an integer T (≤ 100), denoting the number of test cases. Each case contains three integers N (1 ≤ N ≤ 50) , M (1 ≤ M ≤ 50) and Q (1 ≤ Q ≤ 5000). The next Q lines contains an operation defined above. You can assume that there is no invalid reference and **t** is (1≤ t ≤ 10). **If a collision occurs, you have to skip remaining queries of that test case**.

Output:

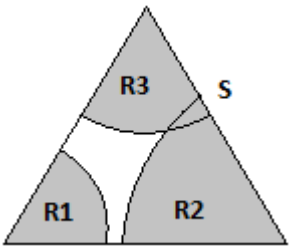
For each case, print the case number first. Then print the full board with snake and fruit, if available, in it. Represent the snake head with '*', its body with '+' and fruit with '$'. **If there are any collision point, you should print head mark where the collision happens**. Each blank cell in the board should be represented with '.'

Sample Input	Sample Output
1 5 5 16 P 1 0 1 M F P 2 0 1 M F M R P 2 3 2 M F M F M F P 3 1 1 M L M L P 0 0 2 M L M F M R	Case 1: $.... ..... .+*.. .++.. .....

Explanation:

Snake start (0,0) position and direction downward.
query P 1 0 1: a fruit reveal at position (1,0)
query M F : snake head goes to (1,0) and increase it's length by 1
query P 2 0 1: a fruit reveal at position (2,0)
query M F: snake head goes to (2,0) and increase it's length by 1
query M R: snake head goes to (2,4)
query P 2 3 2 : a fruit reveal at position (2,3)
query M F: snake head goes to (2,3) and increase it's length by 1
query M F: snake head goes to (2,2)
query M F: snake head goes to (2,1)
query P 3 1 1: a fruit reveal at position (3,1)
query M L: snake head goes to (3,1) and increase it's length by 1
query M L: snake head goes to (3,2)
query P 0 0 2: a fruit reveal at position (0,0)
query M L: snake head goes to (2,2) and collide with it's body, now you should skip rest of the queries this test case and should print current state of the board.

J Trivuj Island



Captain Hanhan is assigned to destroy the Trivuj island with three powerful bombs. Trivuj island is a triangle shaped island whose all sides are equal to **S**. Each bomb has a strength limit. After the explosion of a bomb with strength R, everything within R radius goes under water. Strength of those three bombs are $\mathbf{R_1}$, $\mathbf{R_2}$ and $\mathbf{R_3}$. Captain Hanhan will place the three bombs in three different corners of the Trivuj island. All bombs will explode simultaneously. After completing the mission he has to report about the remaining area of Trivuj island not sunk by the explosions.

As a smart programmer, help Captain Hanhan to find the remaining area of Trivuj island.

Input

Input starts with an integer **T (≤ 1000)**, denoting the number of test cases.

Each case will contain four positive integers S ($1 \le S \le 1000$), R_1, R_2 and R_3 ($1 \le R_1, R_2, R_3 \le 1000$) mentioned earlier.

Output

For each case, print the case number first, then print the remaining area of Trivuj island after the explosion. Errors less than $\mathbf{10^{-4}}$ will be ignored.

Sample Input	Sample Output
3 10 10 10 10 100 10 10 10 100 60 50 30	Case #1: 0.000000 Case #2: 4173.047386 Case #3: 818.417544

K Train Lights

Have you ever travelled by train in Bangladesh? It is (mostly) fun! And though the trains inspired us to write this problem, train-travelling is not a prerequisite to solve this problem.

The trains are made up of several compartments. Each compartment has lights arranged into two columns and **R** rows. There are always some lights that do not work.

The train authority know that it may not possible for them lit all the lights. But they have figured out that the passengers under one faulty light are not dissatisfied if at least three of its neighbouring lights are lit. Two lights are neighbours if they share a common edge.

Now, given **R**, the authority wants you to calculate the minimum number of lights that should work to satisfy all the passengers.

Input:
Input starts with a line containing an integer **T** (≤ 1,000), the number of test cases. Each of following T lines will contain a positive number **R** denoting the number of rows in a compartment. R will fit in a 64-bit signed integer.

Output:
For each test case print a number in a line denoting the minimum number of lights that should work satisfying all the passengers.

Sample Input	Sample Output
2 4 5	6 7