Computer Science Competition
Invitational A 2021
Programming Problem Set

I. General Notes

1. Do the problems in any order you like. They do not have to be done in order from 1 to 12.
2. All problems have a value of 60 points.
3. There is no extraneous input. All input is exactly as specified in the problem. Unless specified by the problem, integer inputs will not have leading zeros. Unless otherwise specified, your program should read to the end of file.
4. Your program should not print extraneous output. Follow the form exactly as given in the problem.
5. A penalty of 5 points will be assessed each time that an incorrect solution is submitted. This penalty will only be assessed if a solution is ultimately judged as correct.

II. Names of Problems

Number	Name
Problem 1	Ana
Problem 2	Bryan
Problem 3	Ethan
Problem 4	Fatimah
Problem 5	Hai
Problem 6	Isaac
Problem 7	Jamari
Problem 8	Kirill
Problem 9	Megan
Problem 10	Oscar
Problem 11	Rebecca
Problem 12	William

1. Ana

Program Name: Ana.java **Input File: None**

Ana is beginning the process of teaching her little sister how to square the numbers between 1 and 12. To square a number, Ana knows that the square of a number is equivalent to multiplying that number by itself. She wants to write a program that will output the product of all numbers between 1 and 12 multiplied by themselves. This way, her little sister can double check her work when Ana is at school. Can you help Ana with this?

Input: There is no input for this problem.

Exact Output:
```
1*1=1
2*2=4
3*3=9
4*4=16
5*5=25
6*6=36
7*7=49
8*8=64
9*9=81
10*10=100
11*11=121
12*12=144
```

2. Bryan

Program Name: Bryan.java **Input File: bryan.dat**

Bryan is having fun picking up summer jobs to earn some spending money. However, he isn't particularly attentive to how much he is earning or how much he's spending. Some weeks, Bryan finds that he's spent more money than he's earned, and his friends and family will soon refuse to stop covering for him.

In order to better keep track of his expenses, Bryan keeps a journal where he writes down how much he makes from each of his jobs and how much he spends each time he goes out. At the end of each week, Bryan totals how much money he's earned and how much money he's spent. Can you help him out by writing a program to do the summations for him?

Input:
The first line is a positive integer W, the number of weeks in this test set. Each week starts with a single positive integer N, the number of transactions in that week. The following N lines each have a string, which describes the transaction, and an integer after it, which is the amount Bryan earned or spent. Positive integers mean Bryan earned that much for completing a job, and negative integers mean that Bryan spent that much.

All integers in the input are non-zero and have absolute value at most 30. The descriptions are strings of no more than 20 lowercase letters.

Output:
For each week, print the case number and:

- If Bryan earned more money than he spent, print "`Wow, Bryan saved $XX`"
- If Bryan spent more than he earned, print "`Oh no! Bryan had to borrow $XX`"
- If Bryan spent exactly as much as he earned, print "`Phew, broke even!`"

Replace XX with the appropriate dollar amount. Do not print any leading zeros, and do not print any cents. Format the answer as in the samples.

Sample Input:
```
3
4
babysitting 10
lemonade 4
game -7
date -7
3
movies -10
mowing 5
carwashing 8
2
tutoring 4
clothes -7
```

Sample Output:
```
Case #1: Phew, broke even!
Case #2: Wow, Bryan saved $3
Case #3: Oh no! Bryan had to borrow $3
```

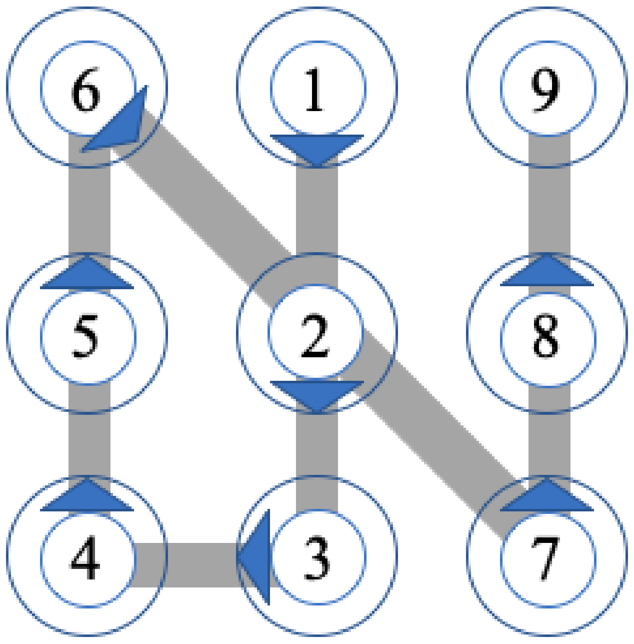
3. Ethan

Program Name: Ethan.java **Input File: ethan.dat**

Ethan has just been hired by United Intercommunication Logistics to test a new unlock pattern idea for their newest smartphone, the U-Cell 20. Their idea is to let the user choose a square grid configuration of numbers representing their password to unlock their cellular device. Research has shown that some users are more likely to remember a shape, or pattern, versus a sequence of numbers. This form of unlock pattern is known as n-pivot unlock pattern, where n is a square of an integer in the range [2,3]. Formally, an unlock pattern is a single stroke that visits each of the four or nine pivots exactly once. The pattern may start at any pivot. It may pass a pivot multiple times but only the first time counts as a visit. The phone unlocks if the pivots are visited in a predefined secret order.

Your task is to compute the distance from 1 -> 2 -> … - > n-1 -> n. The distance between two horizontal points is 1 unit and the distance between two vertical points is 1 unit.

An example, visual configuration for n= 9 is seen below, the total distance would be 9.8284 units.



Input: Input starts with a line containing an integer N (1<=N<=10), the number of test cases. Each test case begins with an integer D (2 <= D <= 3) on a single line, the dimension of the square grid to be used. The following D lines will contain D integers separated by spaces. The D x D integers represent the predefined, secret configuration that would be used to unlock the phone. All D^2 numbers will be used in the password.

Output: For each test case, output “Case # distance: X.XXXX”. Where X.XXXX is the distance from:1 -> 2 -> … -> $D^2 - 1$ -> D^2 rounded to 4 decimal places. For example, for Case 1 the output would be: “`Case 1 distance: 9.8284`”

Sample Input:

```
8
3
6 1 9
5 2 8
4 3 7
2
1 2
3 4
2
1 2
4 3
3
1 2 3
6 5 4
7 8 9
3
1 8 5
7 2 4
6 9 3
3
1 6 8
2 3 4
7 9 5
3
1 2 5
6 3 4
7 8 9
2
1 4
3 2
```

Sample Output:

```
Case 1 distance: 9.8284
Case 2 distance: 3.4142
Case 3 distance: 3.0000
Case 4 distance: 8.0000
Case 5 distance: 12.0711
Case 6 distance: 13.5366
Case 7 distance: 9.2361
Case 8 distance: 3.8284
```

4. Fatimah

Program Name: Fatimah.java **Input File: fatimah.dat**

Fatimah is playing a video game, *Pouch Monsters*. She's amassed a team of N amazing monsters, but not all of them get along. More specifically, there are M pairs of monsters that do not get along and refuse to go adventuring with each other.

Fatimah wants to go adventuring with as many of her monsters as possible. She's asked you to write a computer program to help her out.

Input: The first line of input is T (1 <= T <= 20), the number of test cases. Each test case begins with two integers N (1 <= N <= 12) and M (0 <= M <= 20), where N is the number of monsters, and M is the number of pairs of monsters which do not get along. The following N lines each have the name of one of Fatimah's monsters. The M lines after that each have the names of two of Fatimah's monsters that do not get along.

All monster names are at most 20 lowercase English letters. All monsters listed in a pair are guaranteed to appear in Fatimah's list of monsters. No pair will have the same monster listed twice.

Output: For each test case, output the maximum number of monsters Fatimah can take with her, formatted with the case number as in the samples.

Sample Input:
```
3
3 1
pikawho
charzar
meeth
pikawho meeth
4 2
kaysi
asa
rengo
seju
rengo kaysi
asa seju
5 0
kenny
rayu
chunnley
gail
baisen
```

Sample Output:
```
Case #1: 2
Case #2: 2
Case #3: 5
```

Sample Explanation:
In the first sample, Fatimah can always adventure with "charzar" and can choose to adventure with either "pikawho" or "meeth". Regardless of who she chooses, she can adventure with at most two monsters.

In the third sample, all five monsters get along, so she can adventure with all five of them.

5. Hai

Program Name: Hai.java **Input File: hai.dat**

Hai has just begun to study regular expressions. Hai has a good idea of how regexes work but is always having to look up the various expressions for characters, sets and groups. To make things easier, Hai has decided to write a program that will make a list of examples for some of the regular expressions they have encountered so far. Their program is going to read a line of text from a file and then compare each word in that line to several regular expressions and then print all the words that match that regular expression.

Input: A single space delimited line of text. On the next line there will be a single number N representing the number of regular expressions to check. N will be followed by N lines where each line contains a single regular expression.

Output: The regular expression left justified in ten spaces followed by all the words in the line of text that match that regular expression each separated by a space. If none of the words match the regular expression, print the regular expression as usual followed by "No matches.".

Sample input: *(Note: the line of text will all be on the same line.)*

```
Bill's phone number is 325-456-1234 and his email address is bill@gmail.com.
Let's give him a call ASAP.
3
[a-z]+
\D+
.+m+.+
```

Sample output: *(Note: the output for each regular expression will all be on the same line.)*

```
[a-z]+    phone number is and his email address is give him a call
\D+       Bill's phone number is and his email address is bill@gmail.com.
Let's give him a call ASAP.
.+m+.+    number email bill@gmail.com.
```

6. Isaac

Program Name: Isaac.java **Input File: isaac.dat**

The computer company Isaac works for is introducing a brand new computer line and is developing a new Unix-like operating system to be introduced along with the new computer. Your task is to help Isaac write the formatter for the ls command (which lists directory contents). Input to your program will consist of a list of F filenames that you will sort (ascending based on the ASCII character values) and format into C columns based on the length L of the longest filename. Filenames will be between 1 and 60 (inclusive) characters in length and will be formatted into left-justified columns. The rightmost column will be the width of the longest filename and all other columns will be the width of the longest filename plus 2. There will be as many columns as will fit in 60 characters. **Your program should use as few rows R as possible with columns being filled to capacity from left to right.**

Input: Input starts with a line containing an integer N (1<=N<=10), the number of test cases. Each test case begins with an integer F (1 <= F <= 250) on a single line, the number of files for the given case. The following line will contain F file names, separated by spaces. The filenames are all valid names, but are not guaranteed to be in alphabetical order.

Output: For each test case, output Case # on one line followed by the column indexing for each of the 60 spaces. You are then to output 60 dashes following the column indexing followed by the formatted columns of filenames. The sorted file names 1 to R will be listed down column 1; filenames R + 1 to 2R listed down column 2; etc. For row and column entries where a file name is present, spaces should be output to completely fill the column. For row and column entries that do not have a file name, there should not be any output, i.e., no spaces should be printed. Each case's output is separated by a blank line

Sample Input:

```
5
10
much_longer_name very_long_file_name shorter tiny size-1 size2
12345678.123 mid_size_name 2short4me size3
12
Weaser Alfalfa Stimey Buckwheat Porky Joe Darla Cotton Butch Froggy
Mrs_Crabapple P.D.
26
a b c d e f g h i j k l m n o p q r s t u v w x y z
21
abcdefg bcdefgh cdefghi defghij efghijk fghijkl ghijklm hijklmn
ijklmno jklmnop klmnopq lmnopqr 1111111 mnopqrs nopqrst opqrstu
pqrstuv qrstuvw rstuvwx stuvwxy tuvwxyz
23
Jody jody Buffy bubby sissy Sissy Keith Keith Danny Danny Lori Chris
Shirley Marsha greg Mike Greg jan Bobby Alice Ruben lori mike
```

Sample Output:

```
Case 1
         111111111122222222223333333333444444444455555555556
123456789012345678901234567890123456789012345678901234567890
------------------------------------------------------------
12345678.123         size-1
2short4me            size2
mid_size_name        size3
much_longer_name     tiny
shorter              very_long_file_name

Case 2
         111111111122222222223333333333444444444455555555556
123456789012345678901234567890123456789012345678901234567890
------------------------------------------------------------
Alfalfa        Cotton         Joe            Porky
Buckwheat      Darla          Mrs_Crabapple  Stimey
Butch          Froggy         P.D.           Weaser

Case 3
         111111111122222222223333333333444444444455555555556
123456789012345678901234567890123456789012345678901234567890
------------------------------------------------------------
a  c  e  g  i  k  m  o  q  s  u  w  y
b  d  f  h  j  l  n  p  r  t  v  x  z

Case 4
         111111111122222222223333333333444444444455555555556
123456789012345678901234567890123456789012345678901234567890
------------------------------------------------------------
1111111  defghij  hijklmn  lmnopqr  pqrstuv  tuvwxyz
abcdefg  efghijk  ijklmno  mnopqrs  qrstuvw
bcdefgh  fghijkl  jklmnop  nopqrst  rstuvwx
cdefghi  ghijklm  klmnopq  opqrstu  stuvwxy

Case 5
         111111111122222222223333333333444444444455555555556
123456789012345678901234567890123456789012345678901234567890
------------------------------------------------------------
Alice    Danny    Keith    Mike     bubby    lori
Bobby    Danny    Keith    Ruben    greg     mike
Buffy    Greg     Lori     Shirley  jan      sissy
Chris    Jody     Marsha   Sissy    jody
```

7. Jamari

Program Name: Jamari.java **Input File: jamari.dat**

Jamari has a tiny donut shop in the tiny town of Fayetteville, Texas. Jamari makes fantastic donuts that people come from miles around to buy. Jamari doesn't like to work long hours, so he doesn't make many of his awesome donuts each morning. Consequently, his customers line up early in the morning for a chance to get one of Jamari's world famous donuts. When all the donuts have been sold, Jamari closes his shop and anyone that didn't get a donut has to wait until another day. What we need for you to do today is write a program that will determine who the last person was that got a donut each day.

Input: For each day there will be a list of names and numbers terminated with the string value ">>>". Each name represents a customer. Each number represents how many customers that have been served since the last time a customer was served. When ">>>" is encountered it means Jamari ran out of donuts and closed for the day. The number of days, number of customers each day, and the number of customers served each day is unknown; however, the number of customer names will never be less than the number of customers served and Jamari never has donuts unsold!

Output: For each day, print one of the following:

- The statement "The last person to get a donut was <name>." Where <name> is the name of the last customer to be served followed on a separate line by "<number> customers did not get a donut today." Where <number> represents the number of customers that did not get served.
- If no customers were served, print "No one got a donut today."
- If every customer was served, print "Everyone got a donut today."

Sample input:
```
Melany Jordyn Kaylynn Kyra Kasen Hassan Jaylah 2
Alexandria 1 Leonard Julianna Myles 3 Belinda
Jamari 1 Troy Chase Yasmin Zoey 5 Jaime Jade 4
Brooklyn >>> Tommy Aaliyah Ingrid April 3 Dwayne
Andres River Asa Jaslene 3 >>>
```

Sample output:
```
The last person to get a donut was Yasmin.
4 customers did not get a donut today.
The last person to get a donut was Andres.
3 customers did not get a donut today.
```

8. Kirill

Program Name: Kirill.java **Input File: kirill.dat**

Kirill is learning about self numbers! Let $SOD(n)$ be the sum of the digits in n. For example,

- $SOD(56) = 5 + 6 = 11$
- $SOD(2) = 2$
- $SOD(304) = 3 + 0 + 4 = 7$

A positive integer x is a self number if there is no positive integer y where $y + SOD(y) = x$. 20 is a self number, but 21 is not a self number ($15 + SOD(15) = 15 + 1 + 5 = 21$).

Given an upper bound N, find the largest self number that is less than or equal to N.

Input: The first line of input has an integer T (1 ≤ T ≤ 50), the number of test cases. Each of the next T lines has a single integer N (1 ≤ N ≤ 5,000,000).

Output: For each test case, output the largest self number that is less than or equal to N.

Sample Input:
```
5
22
9
2
60
1234567
```

Sample Output:
```
Case #1: 20
Case #2: 9
Case #3: 1
Case #4: 53
Case #5: 1234557
```

Hint: The first few self numbers are $1, 3, 5, 7, 9, 20, 31, 42,$ and 53.

9. Megan

Program Name: Megan.java **Input File: megan.dat**

Megan has just received multiple messages from alien lifeforms from many different planets! The problem she realizes is that even though these planets use a 26-letter alphabet, each of the planets use a different ordering of letters.

So, contrary to the English alphabet of: abcdefghijklmnopqrstuvwxyz, an example ordering used by one of the planets is: qpotivcaefndxujwzlgrysbhkm. The aliens have sent Megan each of their own respective alphabet orderings. They have also sent lists of words that they need help in determining if the lists are in alphabetical order according to each of their own alphabets. Can you help Megan write a program that will determine this for her?

Input: Input begins with an integer N (1 <= N <= 50), the number of different test cases. The following line contains the alphabetic ordering for that given test case given in lower case letters only. The next line has a list of words separated by commas. All words will consist of lower-case letters only. The number of words will be greater than 0 and less than 100. A blank line will be present to separate test cases.

Output: For each test case, you are to output "Word List # is sorted." if the list is correctly sorted with respect to the given alphabet, or "Word List # is not sorted." otherwise.

Sample Input:
```
8
qpotivcaefndxujwzlgrysbhkm
apple,banana,carrot

qpotivcaefndxujwzlgrysbhkm
carrot,apple,donut,banana

qpotivcaefndxujwzlgrysbhkm
pot,potato,potatoes

qpotivcaefndxujwzlgrysbhkm
pot,potatoes,potato

ndaxyvojzmcrufpbwklhgeistq
nick,sarah,ashley,xray,yellow,violin,open,jelly,zebra,mouse,cat

ndaxyvojzmcrufpbwklhgeistq
name,no,need

ndaxyvojzmcrufpbwklhgeistq
name,names,need,nail

abcdefghijklmnopqrstuvwxyz
apple,apples,banana,bananas,lemon,fruit
```

Sample Output:
```
Word List 1 is not sorted.
Word List 2 is sorted.
Word List 3 is sorted.
Word List 4 is not sorted.
Word List 5 is not sorted.
Word List 6 is sorted.
Word List 7 is not sorted.
Word List 8 is not sorted.
```

10. Oscar

Program Name: Oscar.java **Input File: oscar.dat**

There is a line of ogres moving a big bucket of mud from the swamp to their house. Each ogre carries the bucket until they encounter an ogre that is larger than they are and then hands it off to the larger ogre. There may be several smaller ogres between the ogre that is carrying the mud and the next larger ogre, but those ogres are skipped. If the ogre carrying the bucket comes upon another ogre of the same size, he should pass the bucket. The ogres are not always in the same order and sometimes there are not the same number of ogres available to help out. The first ogre always starts with the bucket of mud. Oscar the ogre oversees keeping track of which ogres carried the mud on each trip. Oscar needs a program to keep track of who carried the mud on each trip.

Input: There will be a single integer N on the first line that represents the number of mud hauling trips that were made. There will then be N lines of names and values separated by a space. The values represent the relative size of each ogre in the line. The lines can each have a different number of ogres.

Output: For each mud trip print which ogres carried the bucket in the order that they carried it.

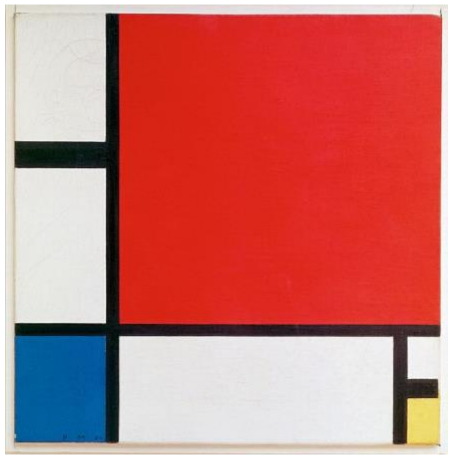
Sample input: *(Note that the third line will all be on the same line in the data file.)*
```
3
Zagut 5 Okork 3 Nogark 6 Zazir 1 Tazir 2 Domuzig 7 Dougurk 3
Treerut 8 Elezor 3 Wazagark 5
Takig 3 Bugrok 2 Krozir 1 Kigruk 4 Wazag 5 Ezigurk 6 Kabekurg 8
Koagurk 1 Uozig 7 Blirag 3
```

Sample output:
```
Zagut Nogark Domuzig
Treerut
Takig Kigruk Wazag Ezigurk Kabekurg
```

11. Rebecca

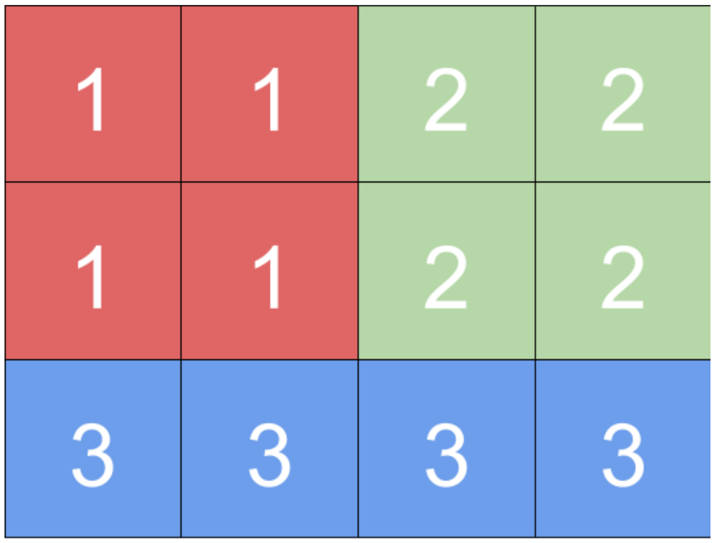
Program Name: Rebecca.java **Input File: rebecca.dat**

Rebecca is a fan of the Dutch art movement *De Stijl*. She especially likes the works of artist Piet Mondrian.



Piet Mondrian, Composition with Red, Blue, and Yellow, 1930.

Many of Mondrian's best known works are made of straight lines and solid colors like the one above. They typically use few colors. Inspired by this art, Rebecca set out to make her own paintings. Here is one of Rebecca's paintings:



1	1	2	2
1	1	2	2
3	3	3	3

Rebecca's canvas is made of R rows, each with C cells. Each cell in her painting has a color, represented by a positive integer. A rectangle is defined using two cells: an upper left cell (r1, c1) and a lower right cell (r2, c2). A rectangle is *monochromatic* if every cell (r, c) where r1 ≤ r ≤ r2 and c1 ≤ c ≤ c2 has the same color. For example, using 1-based indexing for the rows and columns, (1, 1) to (2, 2) is a monochromatic rectangle (all cells have color 1), but (2, 2) to (2, 3) is not (there is a cell of color 1 and a cell of color 2).

Given the layout of Rebecca's painting, find the number of monochromatic rectangles in the painting.

Input: The first line of input has an integer T (1 ≤ T ≤ 20), the number of test cases. Each test case begins with two positive integers R and C, the number of rows and columns in Rebecca's painting. The next R lines each have C space separated integers, the color of each cell of her painting.

The sum of R * C over all test cases will not exceed 6,000,000. Each color in the painting is a positive integer between 1 and 1,000,000,000, both bounds inclusive.

Output: For each test case, output the number of monochromatic rectangles in the grid, formatted with the case number as in the samples.

Sample Input:
```
2
3 4
1 1 2 2
1 1 2 2
3 3 3 3
4 3
8 7 8
5 3 1
2 4 6
6 2 1
```

Sample Output:
```
Case #1: 28
Case #2: 12
```

Sample Explanation: In the first sample, there are 9 rectangles of color 1, 9 rectangles of color 2, and 10 rectangles of color 3, for a total of 28 monochromatic rectangles.
In the second sample, all monochromatic rectangles are 1x1 squares.

12. William

Program Name: William.java **Input File: william.dat**

William is the chairman of the Firethorn High School French Club's recruitment committee. He has been presented with an unsorted list of students that have indicated on a survey that they might like to join the French Club. To make it easier to keep track of who he has contacted and who he has not, William needs an alphabetized list. Since you and your teammates form the French Club's technology committee, the job has fallen to you. Write a program to alphabetize the list of potential new members.

Input: A list of student's names listed first name then last name. There will be an unknown number of names each listed on one line where the first and last names are separated by exactly one space. As luck would have it, there are no duplicate first or last names in the list.

Output: A list of student's names listed first name then last name with exactly one space in between. The list should be alphabetized based on the last name.

Sample input:
```
Carolyn Blair
Jared Hopkins
Maude Almond
Shamas Woodley
Wilfred Blaese
Miya Ridley
Salahuddin Alexander
Lea Richardson
Safwan Campos
Manal Mathews
```

Sample output:
```
Salahuddin Alexander
Maude Almond
Wilfred Blaese
Carolyn Blair
Safwan Campos
Jared Hopkins
Manal Mathews
Lea Richardson
Miya Ridley
Shamas Woodley
```