Part 2: The cat said THAT!

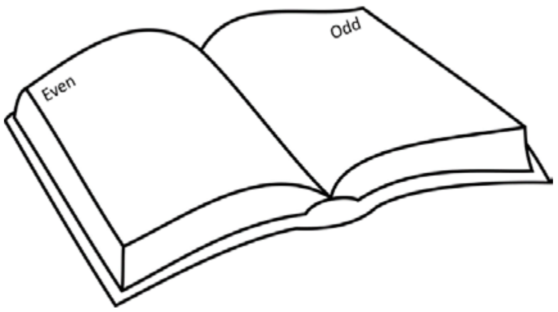
*****The sun did not shine.***
It was too wet to play.
So we sat in the house
All that cold, cold, wet day..... ***The Cat in the Hat, by Dr. Seuss.*

What happens next? Of course, the cat in the hat comes to the house to rid you and Sally of an otherwise, extremely boring evening. Well, he seems to have learnt a good lesson from his last experience when he made such an awful mess! This time he suggests a far more sensible game. This is how it goes:

You open a random page from a book you selected. The total number of pages in the book is either a 2-digit or a 3-digit odd number. The sum of the digits in the page number for the righthand-side page is your score for that turn. **You keep on doing this until the last digit in the righthand-side page number is a 9 or you have taken up 15 turns, whichever happens first.** Your total score is the sum of all the individual scores. Now the other player (Sally) repeats the same process and whoever has the higher total score wins.

Write a function M-file that will simulate this game. The input is the total number of pages (a two or three-digit odd number) in the book you selected to use for the game. There are four outputs for this function. The first two are row vectors that record the page numbers that came up during the play for the two players. The first output is for the player who goes first (`My_run`) and the second output is for the player who goes next (`Sally_run,`). The third output (`winner`) gives information about the winner. It can have one of three values, 0, 1 or 2. The value of 0 means the game is tied, a value of 1 means that you are the winner and a value of 2 means that Sally is the winner. The fourth output (`win_score`) stores the final total score of the winner. It is equal to the tied score if the game is tied.

Generate random integers to simulate the randomly selected page numbers. If the number generated is an even number then consider that as the lefthand-side page number, so that the next number (that will appear on the righthand-side page) will be used as the score-generating page number. This corresponding odd number is also used for checking the termination condition (last digit is 9 or not). If the random number is odd then this is used directly for the score generation and termination condition checking. However, store the exact random number generated (odd or even) as elements in the output arrays.



Note that this means that the presence of both an 8 or 9 at the end of the random integer will serve as a termination condition for a player. Also remember that a player can at most get 15 chances after which he/she must stop even if the last (15th) attempt did not have an 8 or 9 at the end.

A few examples:

```
>> [My_run,Sally_run,winner,win_score]=game(231)

My_run =
   197    42    13    84    64   123    71    71    26    58
Sally_run =
   212    62   166   200   187    49
winner =
   1
win_score =
   97
```

Explanation:
In this case, my total score is:
(1+9+7) + (4+3) + (1+3) + (8+5) + (6+5) + (1+2+3) + (7+1) + (7+1) + (2+7) + (5+9) = 97
Note, that for score calculations the next odd number is used when the random number is even (e.g., (4+3) is the score when random number is 42). However, the even number gets entered in the array, `My_run`.
Also note the last number was 58. The corresponding odd number is 59 (ending in 9) which terminated the game for me.
Sally's score is:
(2+1+3) + (6+3) + (1+6+7) + (2+0+1) + (1+8+7) + (4+9) = 61
As my score (97) is bigger than Sally's score (61), therefore I win the g ame. This information is stores in output `winner=1` and the winning score is stored in the fourth output (`win_score`) of the function.

Example 2

```
>> [My_run,Sally_run,winner,win_score]=game(589)

My_run =
  Columns 1 through 10
   516   470   440    77   457    60    84    83   412   283
  Columns 11 through 15
   130    82   545    51   211
Sally_run =
   422   546   309
winner =
   1
win_score =
   156
```

Explanation:
In this example, I exhausted all my 15 chances. That's why my game was terminated even though on my last (15th) attempt the last digit of my page number is '1' from '211'.
Your code must ensure that each player does not get any more than 15 chances.

Another example:

```
>> [My_run,Sally_run,winner,win_score]=game(565)

My_run =
   394   162   129
Sally_run =
   312   407   240   555   387   272   222   194   412   249
winner =
   2
win_score =
   115
```

Input Restrictions:

The function should halt and return a red error message if the following conditions are not met:

- The input must be a scalar number
- Input must be an odd integer
- The number of digits in the input must be more than 1 and less than 4

Code requirements (must also be followed for passing Cody):

- Use `randi` for generating the random numbers
- The player who plays first must give rise to the first output.,'I'
- The player who plays second must give rise to the second output.,'Sally'

Hints:

For extracting the digits of an integer for score calculations, you may try using functions `fix, rem`.

You may also use other methods you may think of that attains the target of summing over the digits of an integer.

Testing & Submission

Submit to Cody for correctness verification, and publish the following case to a pdf file.

```
>> rng(2131)  %first run this command and then call function
>> [My_run,Sally_run,winner,win_score]=game(493)
```

Upload `game.m` and `game.pdf` to Blackboard