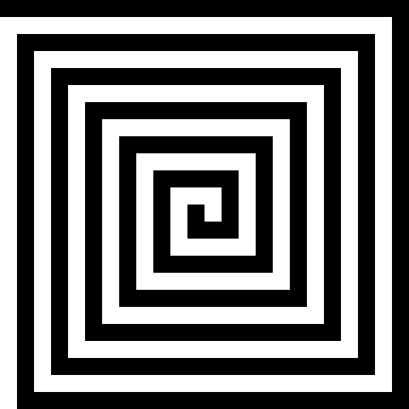
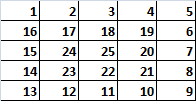
**Problem : Chakravyuha**

During the battle of Mahabharat, when Arjuna was far away in the battlefield, Guru Drona made a Chakravyuha formation of the Kaurava army to capture Yudhisthir Maharaj. Abhimanyu, young son of Arjuna was the only one amongst the remaining Pandava army who knew how to crack the Chakravyuha. He took it upon himself to take the battle to the enemies.   
  
Abhimanyu knew how to get power points when cracking the Chakravyuha. So great was his prowess that rest of the Pandava army could not keep pace with his advances. Worried at the rest of the army falling behind, Yudhisthir Maharaj needs your help to track of Abhimanyu's advances. Write a program that tracks how many power points Abhimanyu has collected and also uncover his trail   
  
A Chakravyuha is a wheel-like formation. Pictorially it is depicted as below



**Fig 1. Chakravyuha**

  
  
A Chakravyuha has a very well-defined co-ordinate system. Each point on the co-ordinate system is manned by a certain unit of the army. The Commander-In-Chief is always located at the center of the army to better co-ordinate his forces. The only way to crack the Chakravyuha is to defeat the units in sequential order.   
  
A Sequential order of units differs structurally based on the radius of the Chakra. The radius can be thought of as length or breadth of the matrix depicted above. The structure i.e. placement of units in *sequential order* is as shown below 

**Fig 2. Army unit placements in Chakravyuha of size 5**

The entry point of the Chakravyuha is always at the (0,0) co-ordinate of the matrix above. This is where the 1st army unit guards. From (0,0) i.e. 1st unit Abhimanyu has to march towards the center at (2,2) where the 25th i.e. the last of the enemy army unit guards. Remember that he has to proceed by destroying the units in sequential fashion. After destroying the first unit, Abhimanyu gets a power point. Thereafter, he gets one after destroying army units which are multiples of 11. You should also be a in a position to tell Yudhisthir Maharaj the location at which Abhimanyu collected his power points. 

**Input Format:**   
  
First line of input will be length as well as breadth of the army units, say N

**Output Format:**

* Print NxN matrix depicting the placement of army units, with unit numbers delimited by (\t) Tab character
* Print Total power points collected
* Print coordinates of power points collected in sequential fashion (one per line)

**Constraints:**

1. **0 < N <=100**

**Sample Input and Output**

|  |  |  |
| --- | --- | --- |
| **SNo.** | **Input** | **Output** |
| 1 | 2 | 1 2 4 3 Total Power points : 1 (0,0) |
| 2 | 5 | 1 2 3 4 5 16 17 18 19 6 15 24 25 20 7 14 23 22 21 8 13 12 11 10 9 Total Power points : 3 (0,0) (4,2) (3,2) |

**Problem : The Vita Sum**

Tom the cat is brushing up his Math skills. He has a bag containing N balls of different colors. Now Tom can randomly pick any even number of balls from the bag. Tom wants to find out the sum of all such combinations of balls that he can pull out from the bag. Given he can pull out at max K balls in one pick.

**Input Format:**   
  
First line contains two space separated numbers N and K

**Output Format:**   
  
The output is the sum of all the combinations of balls he can pull out modulo 10^9+7 i.e. (1000000007)

**Constraints:**

1. **0<=N,k<=10^9**
2. **N >= k**

**Sample Input and Output**

|  |  |  |  |
| --- | --- | --- | --- |
| **SNo.** | **Input** | **Output** | **Explaination** |
| 1 | 4 4 | 8 | We need 4C0 + 4C2+ 4C4= 1+6+1=8 |
| 2 | 8 3 | 29 | We need 8C0 + 8C2= 1+28=29 |

**Problem : Verify JSON Object validity**

A JSON object is a key-value pair data structure that is enclosed within { }. A sample JSON object would look like  
{  
"key1":"value1",  
"key2":"value2",  
"key3": {   
"key4":"value4",  
"key5":"value5"}  
"key6":"value6",  
"key7":[  
{  
"key8":"value8"  
}]  
}  
  
Given a JSON object, ignore the literal values of the object and check whether the distinguishing characters and notation of the object are valid to determine if the JSON object is valid or not.   
  
Note:

1. Key3 points to another JSON object (Concept of nesting of JSON objects).
2. Key7 points to an array of JSON objects.

You may wish to refer <http://json.org/>  to get a more formal description of JSON grammar. <http://www.tutorialspoint.com/json/json_overview.htm> are also good resources to understand JSON specifications. 

**Input Format:**

1. First line contains a pattern of JSON without any literal

**Output Format:**   
  
Print 1 if pattern is valid, -1 otherwise.

**Constraints:**

1. **A JSON object should start with '{' and ends with a '}'.**
2. **The key and value should be separated by a ':'.**
3. **A ',' suggests an additional JSON property.**
4. **An array only consists of JSON objects. It cannot contain a "key":"value" pair by itself.**

**Example 1:**   
  
**Input**   
{:[{},{}]}   
  
**Outputput**   
1   
  
**Explanation**   
{  
"Key": [{  
"Key": "Value"  
}, {  
"Key": "Value"  
}]  
}  
Pattern is following all constrains hence prints 1

**Example 2:**   
  
**Input**   
{:{[]},{}}   
  
**Outputput**   
-1   
  
**Explanation**   
Covert this pattern in a JSON Object  
  
{  
"Key": {  
[  
"Key": "Value"  
]  
},  
{  
"Key": "Value"  
}  
}  
  
Constraint 4 "An array only consists of JSON objects. It cannot contain a "key":"value" pair by itself." not followed here, so it's a invalid pattern, hence prints -1

**Problem : Ram and Chocolates**

Ram is a school boy. He has n good friends numbered from 1 to n. Let this be denoted by an Array A. The friend with higher array index is closer to Ram. It's his birthday today and hence he wants to distribute his birthday chocolates in a way such that A[i+1]=A[i]+3. Now he had n boxes numbered from 1 to n for each of his friend and his mother had filled them with some random number of chocolates, However Ram has made up his mind to distribute the the chocolates in the way described above. He wants to know if it is possible to re-arrange the chocolates in the boxes to have a distribution pattern he likes.

**Input Format:**

1. First line contains the number n denoting the number of boxes
2. Second line contains n space separated numbers denoting chocolates in each box.

**Output Format:**

* Output **Yes** if the rearrangement is possible followed by the minimum number of steps required to rearrange, delimited by whitespace
* Output **No** if such rearrangement is not possible.

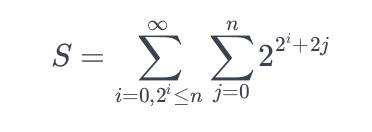
**Constraints:**

1. **1<=n<=25**
2. **1<=Ai<=500**

**Sample Input and Output**

|  |  |  |  |
| --- | --- | --- | --- |
| **SNo.** | **Input** | **Output** | **Explaination** |
| 1 | 3 4 8 10 | No |  |
| 2 | 5 6 10 11 16 17 | Yes 2 | Step1: 6 9 12 16 17 Step 2: 6 9 12 15 18 |

**Problem : Last Digit**

**Statement :**   
  
Compute the last digit of the following sequence   
  
   
  
i.e. Summation of F(n) from i = 0 to 2^i &tl;= n, where F(n) is the summation of 2^(2^i+2j) Where j varies from 0 to n

**Input Format:**   
  
Single integer representing n

**Output Format:**   
  
Last digit of the number S

**Constraints:**

1. **0<=n<=10^17**

**Sample Input and Output**

|  |  |  |
| --- | --- | --- |
| **SNo.** | **Input** | **Output** |
| 1 | 3 | 0 |
| 2 | 10 | 8 |

***Note***:

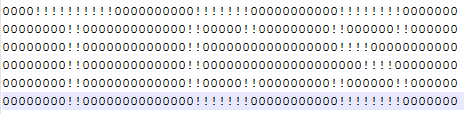
**Problem : RLE String**

Given a Run Length Encoded ASCII (Base 64) String as input, write a program to decode the string and print the corresponding image. Number of characters in each row of the resulting image is always 57.   
  
This question can be understood once we understand how RLE maps to an image. Have a look at the Example section to get this understanding.

**Input Format:**   
  
RLE ASCII String

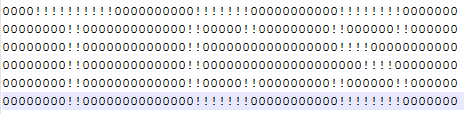
**Output Format:**   
  
Ascii data resembling an image corresponding to input RLE String

**Example:**   
  
The following string is a RLE string   
  
**DJJGKHG HBMBEBIBFBF HBMBQDK HBMBTDH HBMBEBIBFBF HBNGKHG**   
  
It gets converted into the image below. You should see 'TCS' emerge by embossing the '!' character on '0' character

  
**Fig. 1. Ascii Image corresponding to RLE string shown above**

Note: - Each row comprising only of '!' and '0' has a length of exactly 57 characters   
Now let's understand the steps to convert the RLE into the image

1. Split the RLE string on space character. For each substring achieved after splitting do step 2
2. **Lets perform the required calculation on first row of ascii image**:- which is **DJJGKHG**
   1. Where, highlighted value is calculated as (Ascii value of(D)-64)ïƒ¨ (68-64)=4
   2. (Ascii value of(J)-64)->(74-64)=10
   3. (Ascii value of(J)-64)-> (74-64)=10
   4. (Ascii value of(G)-64)-> (71-64)=7
   5. (Ascii value of(K)-64)-> (75-64)=11
   6. (Ascii value of(H)-64)-> (72-64)=8
   7. (Ascii value of(G)-64)-> (71-64)=7
3. Addition of all numerical values for each character in every sub-string as computed in step 2 should equal 57
4. Corresponding to each alphabet in the sub-string, say D, a number, 4 in this case, is achieved by operation denoted in step 2
5. Print '0' 4 times (let's call this operation multiply) which will produce a string '0000'
6. Take the next character, J in this case and number 10 achieved by performing operation denoted in step 2
7. Toggle '0' to '!' and print it 10 times which will produce a string '!!!!!!!!!!'
8. Similarly, for each even index in sub-string multiply '0' with corresponding number and for each odd index in sub-string multiply '!' with corresponding number. Sub-string starts with index 0.
9. After processing each sub-string you will get the corresponding row in the ascii image
10. When all sub-strings are processed you will get all rows which will form the ascii image. Print that as output.

**Sample Input and Output**  
  
**Input1:**   
DJJGKHG HBMBEBIBFBF HBMBQDK HBMBTDH HBMBEBIBFBF HBNGKHG   
  
**Output1:**  
  
   
  
**Input2:**   
YFZ YFZ YFZ YFZ JdK JdK JdK YFZ YFZ YFZ YFZ YFZ   
  
**Output1:**  
