**Problem1:** An extra day is added to the calendar almost every four years as February 29, and the day is called a *leap day*. It corrects the calendar for the fact that our planet takes approximately 365.25 days to orbit the sun. A leap year contains a leap day.

In the Gregorian calendar, three conditions are used to identify leap years:

* The year can be evenly divided by 4, is a leap year, unless:
  + The year can be evenly divided by 100, it is NOT a leap year, unless:
    - The year is also evenly divisible by 400. Then it is a leap year.

This means that in the Gregorian calendar, the years 2000 and 2400 are leap years, while 1800, 1900, 2100, 2200, 2300 and 2500 are NOT leap years.

**Task**

Given a year, determine whether it is a leap year. If it is a leap year, return the Boolean True, otherwise return False.

Note that the code stub provided reads from STDIN and passes arguments to the is\_leap function. It is only necessary to complete the is\_leap function.

**Input Format**

Read **year** the year to test.

**Constraints : 1990 <= year <= 10^5**

**Output Format**

The function must return a Boolean value (True/False). Output is handled by the provided code stub

**Input (STDIN) : 1990**

**Expected Output : False**

def is\_leap(year):

    leap = False

   # Write your logic here

    if year % 4 == 0:

        if year % 100 !=0 or year % 400 == 0:

            leap = True

    return leap

year = int(input())

print(is\_leap(year))

…………………………………………………………………………………………………………………………………………………………………………………

**Problem2:** The included code stub will read an integer n from STDIN. Without using any string methods, try to print the 123………n

Note that "…." represents the consecutive values in between.

**Example** n = 5

**Print the string**  12345

**Input Format**

The first line contains an integer n

**Constraints**

**1 < =** n<=150

**Output Format**

Print the list of integers from 1  through n as a string, without spaces.

**Expected Output : 123**

if \_\_name\_\_ == '\_\_main\_\_':

    n = int(input())

for i in range(1,n+1):

    print(i,end="")

**Problem3:** In Python, a string of text can be aligned *left, right* and *center*.

**.ljust(width)**

This method returns a left aligned string of length *width*.

>>> width = 20

>>> print 'HackerRank'.ljust(width,'-')

HackerRank----------

**.center(width)**

This method returns a centered string of length *width*.

>>> width = 20

>>> print 'HackerRank'.center(width,'-')

-----HackerRank-----

**.rjust(width)**

This method returns a right aligned string of length *width*.

>>> width = 20

>>> print 'HackerRank'.rjust(width,'-')

----------HackerRank

**Task**

You are given a partial code that is used for generating the *HackerRank Logo* of variable *thickness*.  
Your task is to replace the blank (\_\_\_\_\_\_) with *rjust, ljust* or *center*.

**Input Format**

A single line containing the *thickness* value for the logo.

**Constraints**

The *thickness* must be an *odd* number.  
**Output Format**

Output the desired logo.

**Sample Input**

5

**Sample Output**

H

HHH

HHHHH

HHHHHHH

HHHHHHHHH

HHHHH HHHHH

HHHHH HHHHH

HHHHH HHHHH

HHHHH HHHHH

HHHHH HHHHH

HHHHH HHHHH

HHHHHHHHHHHHHHHHHHHHHHHHH

HHHHHHHHHHHHHHHHHHHHHHHHH

HHHHHHHHHHHHHHHHHHHHHHHHH

HHHHH HHHHH

HHHHH HHHHH

HHHHH HHHHH

HHHHH HHHHH

HHHHH HHHHH

HHHHH HHHHH

HHHHHHHHH

HHHHHHH

HHHHH

HHH

H

# Replace the blank (\_\_\_\_\_\_) with rjust, ljust or center

thickness = int(input().strip())

c = 'H'

# Top Cone

for i in range(thickness):

    print((c\*i).rjust(thickness-1)+c+(c\*i).ljust(thickness-1))

# Top Pillars

for i in range(thickness+1):

    print((c\*thickness).center(thickness\*2)+(c\*thickness).center(thickness\*6))

# Middle Belt

for i in range((thickness+1)//2):

    print((c\*thickness\*5).center(thickness\*6))

# Bottom Pillars

for i in range(thickness+1):

    print((c\*thickness).center(thickness\*2)+(c\*thickness).center(thickness\*6))

# Bottom Cone

for i in range(thickness):

    print(((c\*(thickness-i-1)).rjust(thickness)+c+(c\*(thickness-i-1)).ljust(thickness)).rjust(thickness\*6))

…………………………………………………………………………………………………………………………………………………………………………………

**problem5:** You are given a string s  and width w. Your task is to wrap the string into a paragraph of width .

**Function Description**

Complete the *wrap* function in the editor below.

*wrap* has the following parameters:

* ***string string****:* a long string
* ***int max\_width****:* the width to wrap to

**Returns**

* *string:* a single string with newline characters ('\n') where the breaks should be

**Input Format**

The first line contains a string, **string**.  
The second line contains the width, *max\_width*.

**Constraints**

0 < len(string) < 1000

0 < max\_width < len(string)

**Sample Input:**

ABCDEFGHIJKLIMNOQRSTUVWXYZ

4

**Sample Output:**

ABCD

EFGH

IJKL

IMNO

QRST

UVWX

YZ

import textwrap

def wrap(string, max\_width):

    wrapped\_string = textwrap.fill(string, max\_width)

    return wrapped\_string

if \_\_name\_\_ == '\_\_main\_\_':

    string, max\_width = input(), int(input())

    result = wrap(string, max\_width)

    print(result)

…………………………………………………………………………………………………………………………………………………………………………………

**Problem6:** Mr. Vincent works in a door mat manufacturing company. One day, he designed a new door mat with the following specifications:

* Mat size must be N x M. ( N is an odd natural number, and M is 3  times .)
* The design should have 'WELCOME' written in the center.
* The design pattern should only use **|**, **.** and **-** characters.

**Sample Designs**

Size: 7 x 21

---------.|.---------

------.|..|..|.------

---.|..|..|..|..|.---

-------WELCOME-------

---.|..|..|..|..|.---

------.|..|..|.------

---------.|.---------

Size: 11 x 33

---------------.|.---------------

------------.|..|..|.------------

---------.|..|..|..|..|.---------

------.|..|..|..|..|..|..|.------

---.|..|..|..|..|..|..|..|..|.---

-------------WELCOME-------------

---.|..|..|..|..|..|..|..|..|.---

------.|..|..|..|..|..|..|.------

---------.|..|..|..|..|.---------

------------.|..|..|.------------

---------------.|.---------------

**Input Format**

A single line containing the space separated values of N  and M

**Constraints**

5 < N < 101

15 < M < 303

**Output Format**

Output the design pattern.

#Enter your code here. Read input from STDIN.Print output to STDOUT

def print\_door\_mat(n, m):

    pattern = [('.|.'\*(2\*i + 1)).center(m, '') for i in range(n//2]

    welcome = 'WELCOME'.center(m, '-')

    door\_mat = pattern + [welcome] + pattern[::-1]

    for line in door\_mat:

        print(line)

if \_\_name\_\_ == '\_\_main\_\_':

    n, m = map(int, input().split())

    print\_door\_mat(n, m)

…………………………………………………………………………………………………………………………………………………………………………………

**Problem7:** Given an integer n,  print the following values for each integer  from 1 to n:

1. Decimal
2. Octal
3. Hexadecimal (capitalized)
4. Binary

**Function Description**

Complete the *print\_formatted* function in the editor below.

*print\_formatted* has the following parameters:

* *int number:* the maximum value to print

**Prints**

The four values must be printed on a single line *in the order specified above* for each i from 1 to number . Each value should be space-padded to match the width of the *binary* value of  and the values should be separated by a single space.

**Input Format**

A single integer denoting n

**Constraints**

1 <=n<=99

**Sample Input**

17

**Sample Output**

1 1 1 1

2 2 2 10

3 3 3 11

4 4 4 100

5 5 5 101

6 6 6 110

7 7 7 111

8 10 8 1000

9 11 9 1001

10 12 A 1010

………………………………………………………………………………………………………

def print\_formatted(number):

    # your code goes here

    width = len(bin(number)[2:])

    for i in range(1, number+1):

        decimal = str(i)

        octal = oct(i)[2:]

        hexadecimal = hex(i)[2:].upper()

        binary = bin(i)[2:]

        print(decimal.rjust(width), octal.rjust(width),

hexadecimal.rjust(width), binary.rjust(width))

if \_\_name\_\_ == '\_\_main\_\_':

    n = int(input())

    print\_formatted(n)

…………………………………………………………………………………………………………………………………………………………………………………

**Problem8:** You are given an integer N. Your task is to print an alphabet rangoli of size N .

(Rangoli is a form of Indian folk art based on creation of patterns.)

Different sizes of alphabet rangoli are shown below:

#size 3

----c----

--c-b-c--

c-b-a-b-c

--c-b-c--

----c----

#size 5

--------e--------

------e-d-e------

----e-d-c-d-e----

--e-d-c-b-c-d-e--

e-d-c-b-a-b-c-d-e

--e-d-c-b-c-d-e--

----e-d-c-d-e----

------e-d-e------

--------e--------

#size 10

------------------j------------------

----------------j-i-j----------------

--------------j-i-h-i-j--------------

------------j-i-h-g-h-i-j------------

----------j-i-h-g-f-g-h-i-j----------

--------j-i-h-g-f-e-f-g-h-i-j--------

------j-i-h-g-f-e-d-e-f-g-h-i-j------

----j-i-h-g-f-e-d-c-d-e-f-g-h-i-j----

--j-i-h-g-f-e-d-c-b-c-d-e-f-g-h-i-j--

j-i-h-g-f-e-d-c-b-a-b-c-d-e-f-g-h-i-j

--j-i-h-g-f-e-d-c-b-c-d-e-f-g-h-i-j--

----j-i-h-g-f-e-d-c-d-e-f-g-h-i-j----

------j-i-h-g-f-e-d-e-f-g-h-i-j------

--------j-i-h-g-f-e-f-g-h-i-j--------

----------j-i-h-g-f-g-h-i-j----------

------------j-i-h-g-h-i-j------------

--------------j-i-h-i-j--------------

----------------j-i-j----------------

------------------j------------------

The center of the rangoli has the first alphabet letter *a*, and the boundary has the  alphabet letter (in alphabetical order).

**Function Description**

Complete the *rangoli* function in the editor below.

*rangoli* has the following parameters:

* *int size:* the size of the rangoli

**Returns**

* *string:* a single string made up of each of the lines of the rangoli separated by a newline character (\n)

**Input Format**

Only one line of input containing , the size of the rangoli

……………………………………………………………………………………………….

**Problem9**: **Task** Perform *append*, *pop*, *popleft* and *appendleft* methods on an empty deque .

**Input Format**

The first line contains an integer , the number of operations.  
The next  lines contains the space separated names of methods and their values.

**Constraints**

**Output Format**

Print the space separated elements of deque .

**Sample Input**

6

append 1

append 2

append 3

appendleft 4

pop

popleft

**Sample Output**

1 2

……………………………………………………………………………………………………..

Problem10:

.