**HackThread #1.**

**Objective**  
In this challenge, we're going to use loops to help us do some simple math. Check out the [Tutorial](https://www.hackerrank.com/challenges/30-loops/tutorial) tab to learn more.

**Task**  
Given an integer,n , print its first  10 multiples. Each multiple n x i (where 1≤i≤10 ) should be printed on a new line in the form: n x i = result.

**Input Format**

A single integer, n .

**Constraints**

* 2≤n≤20

**Output Format**

Print 10 lines of output; each line i (where1≤i≤10  ) contains the  of  the result n x i in the form:  
n x i = result.

**Sample Input**

2

**Sample Output**

2 x 1 = 2

2 x 2 = 4

2 x 3 = 6

2 x 4 = 8

2 x 5 = 10

2 x 6 = 12

2 x 7 = 14

2 x 8 = 16

2 x 9 = 18

2 x 10 = 20

**HackThread #2.**

**Objective**  
Today, we're learning about the *Array* data structure. Check out the [Tutorial](https://www.hackerrank.com/challenges/30-arrays/tutorial) tab for learning materials and an instructional video!

**Task**  
Given an array,A , of  N integers, print A 's elements in *reverse* order as a single line of space-separated numbers.

**Input Format**

The first line contains an integer,  N(the size of our array).  
The second line contains  N space-separated integers describing array  A's elements.

**Constraints**

* 1≤ n ≤1000
* 1≤ Ai ≤10000, where Ai is the ith integer in the array

**Output Format**

Print the elements of array A in reverse order as a single line of space-separated numbers.

**Sample Input**

4

1 4 3 2

**Sample Output**

2 3 4 1

**HackThread #3.**

You are given a string and your task is to *swap cases*. In other words, convert all lowercase letters to uppercase letters and vice versa.

**For Example:**

Www.HackerRank.com → wWW.hACKERrANK.COM

Pythonist 2 → pYTHONIST 2

**Input Format**

A single line containing a string S.

**Constraints**

**0<len(S)≤1000**

**Output Format**

Print the modified string S.

**Sample Input 0**

HackerRank.com presents "Pythonist 2".

**Sample Output 0**

hACKERrANK.COM PRESENTS "pYTHONIST 2".

**HackThread #4.**

In Python, a string can be split on a delimiter.

**Example:**

>>> a = "this is a string"

>>> a = a.split(" ") # a is converted to a list of strings.

>>> print a

['this', 'is', 'a', 'string']

Joining a string is simple:

>>> a = "-".join(a)

>>> print a

this-is-a-string

**Task**  
You are given a string. Split the string on a " " (space) delimiter and join using a - hyphen.

**Input Format**  
The first line contains a string consisting of space separated words.

**Output Format**  
Print the formatted string as explained above.

**Sample Input**

this is a string

**Sample Output**

this-is-a-string

**HackThread #5.**

You are given the firstname and lastname of a person on two different lines. Your task is to read them and print the following:

Hello firstname lastname! You just delved into python.

**Input Format**

The first line contains the first name, and the second line contains the last name.

**Constraints**

The length of the first and last name ≤ 10.

**Output Format**

Print the output as mentioned above.

**Sample Input 0**

Ross

Taylor

**Sample Output 0**

Hello Ross Taylor! You just delved into python.

**Explanation 0**

The input read by the program is stored as a string data type. A string is a collection of characters.

**HackThread #6.**

We have seen that lists are mutable (they can be changed), and tuples are immutable (they cannot be changed).

Let's try to understand this with an example.

You are given an immutable string, and you want to make changes to it.

**Example**

>>> string = "abracadabra"

You can access an index by:

>>> print string[5]

a

What if you would like to assign a value?

>>> string[5] = 'k'

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

TypeError: 'str' object does not support item assignment

How would you approach this?

* One solution is to convert the string to a list and then change the value.

**Example**

>>> string = "abracadabra"

>>> l = list(string)

>>> l[5] = 'k'

>>> string = ''.join(l)

>>> print string

abrackdabra

* Another approach is to slice the string and join it back.

Example

>>> string = string[:5] + "k" + string[6:]

>>> print string

abrackdabra

**Task**  
Read a given string, change the character at a given index and then print the modified string.

**Input Format**  
The first line contains a string, S .  
The next line contains an integer i , denoting the index location and a character  c separated by a space.

**Output Format**  
Using any of the methods explained above, replace the character at index i with character c .

**Sample Input**

abracadabra

5 k

**Sample Output**

abrackdabra

**HackThread #7.**

In this challenge, the user enters a string and a substring. You have to print the number of times that the substring occurs in the given string. String traversal will take place from left to right, not from right to left.

**NOTE:** String letters are case-sensitive.

**Input Format**

The first line of input contains the original string. The next line contains the substring.

**Constraints**

1≤len(string)≤200  
Each character in the string is an *ascii* character.

**Output Format**

Output the integer number indicating the total number of occurrences of the substring in the original string.

**Sample Input**

ABCDCDC

CDC

**Sample Output**

2

**Concept**

Some string processing examples, [such as these](http://www.thelearningpoint.net/computer-science/learning-python-programming-and-data-structures/learning-python-programming-and-data-structures--tutorial-12--string-manipulation), might be useful.  
There are a couple of new concepts:  
In Python, the length of a string is found by the function len(s), where  is the string.  
To traverse through the length of a string, use a *for* loop:

for i in range(0, len(s)):

print (s[i])

A range function is used to loop over some length:

range (0, 5)

Here, the range loops over  0 to 4. 5  is excluded.

**HackThread #8.**

Python has built-in string validation methods for basic data. It can check if a string is composed of alphabetical characters, alphanumeric characters, digits, etc.

[str.isalnum()](https://docs.python.org/2/library/stdtypes.html#str.isalnum)  
This method checks if all the characters of a string are alphanumeric *(a-z, A-Z and 0-9)*.

>>> print 'ab123'.isalnum()

True

>>> print 'ab123#'.isalnum()

False

[str.isalpha()](https://docs.python.org/2/library/stdtypes.html#str.isalpha)  
This method checks if all the characters of a string are alphabetical *(a-z and A-Z)*.

>>> print 'abcD'.isalpha()

True

>>> print 'abcd1'.isalpha()

False

[str.isdigit()](https://docs.python.org/2/library/stdtypes.html#str.isdigit)  
This method checks if all the characters of a string are digits *(0-9)*.

>>> print '1234'.isdigit()

True

>>> print '123edsd'.isdigit()

False

[str.islower()](https://docs.python.org/2/library/stdtypes.html#str.islower)  
This method checks if all the characters of a string are lowercase characters *(a-z)*.

>>> print 'abcd123#'.islower()

True

>>> print 'Abcd123#'.islower()

False

[str.isupper()](https://docs.python.org/2/library/stdtypes.html#str.isupper)  
This method checks if all the characters of a string are uppercase characters *(A-Z)*.

>>> print 'ABCD123#'.isupper()

True

>>> print 'Abcd123#'.isupper()

False

**Task**

You are given a string S.  
Your task is to find out if the string S contains: *alphanumeric characters, alphabetical characters, digits, lowercase and uppercase characters*.

**Input Format**

A single line containing a string S .

**Constraints**

**0≤len(S)≤1000**

**Output Format**

In the first line, print True if S has any *alphanumeric characters*. Otherwise, print False.  
In the second line, print True if S has any *alphabetical characters*. Otherwise, print False.  
In the third line, print True if S has any *digits*. Otherwise, print False.  
In the fourth line, print True if S has any *lowercase characters*. Otherwise, print False.  
In the fifth line, print True if S has any *uppercase characters*. Otherwise, print False.

**Sample Input**

qA2

**Sample Output**

True

True

True

True

True

**HackThread #9.**

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.  
0 <thickness<50

**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

**HackThread #10**

Consider a list (list = []). You can perform the following commands:

1. insert i e: Insert integer  e at position i.
2. print: Print the list.
3. remove e: Delete the first occurrence of integer e.
4. append e: Insert integer e at the end of the list.
5. sort: Sort the list.
6. pop: Pop the last element from the list.
7. reverse: Reverse the list.

Initialize your list and read in the value of n followed by  n lines of commands where each command will be of the 7 types listed above. Iterate through each command in order and perform the corresponding operation on your list.

**Example**  
N = 4

append 1

append 2

insert 3 1

print

* append 1: Append 1 to the list , arr = [1].
* append 2: Append 1 to the list , arr = [1,2].
* append 3: Append 1 to the list , arr = [1,3,2].

print:Print the array  
Output:

[1, 3, 2]

**Input Format**

The first line contains an integer,n , denoting the number of commands.  
Each line  i of the n subsequent lines contains one of the commands described above.

**Constraints**

* The elements added to the list must be *integers*.

**Output Format**

For each command of type print, print the list on a new line.

**Sample Input 0**

12

insert 0 5

insert 1 10

insert 0 6

print

remove 6

append 9

append 1

sort

print

pop

reverse

print

**Sample Output 0**

[6, 5, 10]

[1, 5, 9, 10]

[9, 5, 1]

**HackThread #11**

**Task**  
Given an integer,n, and n  space-separated integers as input, create a tuple,t , of those n  integers. Then compute and print the result of hash(t).

**Note:** [hash()](https://docs.python.org/3/library/functions.html#hash) is one of the functions in the \_\_builtins\_\_ module, so it need not be imported.

**Input Format**

The first line contains an integer,n, denoting the number of elements in the tuple.  
The second line contains n space-separated integers describing the elements in tuple t.

**Output Format**

Print the result of hash(t).

**Sample Input 0**

2

1 2

**Sample Output 0**

3713081631934410656

**HackThread #12**

The provided code stub will read in a dictionary containing key/value pairs of name:[marks] for a list of students. Print the average of the marks array for the student name provided, showing 2 places after the decimal.

**Input Format**

The first line contains the integer n , the number of students' records. The next n lines contain the names and marks obtained by a student, each value separated by a space. The final line contains **query\_name**, the name of a student to query.

**Constraints**

* 2≤n≤10
* 0≤marks[i]≤100
* Length of marks arrays=3

**Output Format**

Print one line: The average of the marks obtained by the particular student correct to 2 decimal places.

**Sample Input 0**

3

Krishna 67 68 69

Arjun 70 98 63

Malika 52 56 60

Malika

**Sample Output 0**

56.00

**Explanation 0**

Marks for Malika are{52,56,60} whose average is (52+56+60)/3 =>56

**Sample Input 1**

2

Harsh 25 26.5 28

Anurag 26 28 30

Harsh

**Sample Output 1**

26.50