1. Write a function that takes a list of lists and returns the value of all of the symbols in it, where each symbol adds or takes something from the total score. Symbol values:

# = 5
O = 3
X = 1
! = -1
!! = -3
!!! = -5

A list of lists containing 2 #s, a O, and a !!! would equal (0 + 5 + 5 + 3 - 5) 8.

If the final score is negative, return 0 (e.g. 3 #s, 3 !!s, 2 !!!s and a X would be (0 + 5 + 5 + 5 - 3 - 3 - 3 - 5 - 5 + 1) -3, so return 0.

Examples

```
check_score([
  ["#", "!"],
  ["!!", "X"]
])    2
```

```
check_score([
  ["!!!", "O", "!"],
  ["X", "#", "!!!"],
  ["!!", "X", "O"]
])    0
```

2. Create a function that takes a variable number of arguments, each argument representing the number of items in a group, and returns the number of permutations (combinations) of items that you could get by taking one item from each group.

Examples

```
combinations(2, 3)    6

combinations(3, 7, 4)    84

combinations(2, 3, 4, 5)    120
```

3. Create a function that takes a string as an argument and returns the Morse code equivalent.

Examples

encode_morse("EDABBIT CHALLENGE") ". -.. .- -... -... .. - -.-. .... .- .-.. .-.. . -. --. ."

encode_morse("HELP ME !") ".... . .-.. .--. -- . -.-.--"

This dictionary can be used for coding:

```
char_to_dots = {
  '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': '--..', ' ': ' ', '0': '-----',
  '1': '.----', '2': '..---', '3': '...--', '4': '....-', '5': '.....',
  '6': '-....', '7': '--...', '8': '---..', '9': '----.',
  '&': '.-...', "'": '.----.', '@': '.--.-.', ')': '-.--.-', '(': '-.--.',
  ':': '---...', ',': '--..--', '=': '-...-', '!': '-.-.--', '.': '.-.-.-',
  '-': '-....-', '+': '.-.-.', '"': '.-..-.', '?': '..--..', '/': '-..-.'
}
```

4. Write a function that takes a number and returns True if it's a prime; False otherwise. The number can be 2^64-1 (2 to the power of 63, not XOR). With the standard technique it would be O(2^64-1), which is much too large for the 10 second time limit.

Examples

prime(7) True

prime(56963) True

prime(5151512515524) False

5. Create a function that converts a word to a bitstring and then to a boolean list based on the following criteria:

1. Locate the position of the letter in the English alphabet (from 1 to 26).
2. Odd positions will be represented as 1 and 0 otherwise.
3. Convert the represented positions to boolean values, 1 for True and 0 for False.
4. Store the conversions into an array.

Examples

to_boolean_list("deep") [False, True, True, False]
deep converts to 0110
d is the 4th alphabet - 0
e is the 5th alphabet - 1
e is the 5th alphabet - 1
p is the 16th alphabet - 0

to_boolean_list("loves") [False, True, False, True, True]

to_boolean_list("tesh") [False, True, True, False]