سوال ۱: تابعی بنویسید که عددی بین 0 تا 30 را به عنوان آرگومان ورودی دریافت کند. سپس مقسوم علیه های اول آن را محاسبه کرده و در خروجی چاپ کند.

Example:
primeDivisors(20) → [2, 5]
primeDivisors(24) → [2, 3]
primeDivisors(13) → [13]

سوال ۲: تابعی بنویسید که کلمه ای را به عنوان آرگومان ورودی بگیرد و حرفی که بالاترین ایندکس را در حروف الفبا دارد به همراه عدد ایندکس در خروجی برگرداند. به شکل یک داده متنی شامل شماره ایندکس و نام حرف.

آرایه ای از حروف الفبا:

const alphabet = ["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"]

Example:
alphabetIndex("Flavio") → 22v
alphabetIndex("Andrey") → 25y
alphabetIndex("Oscar") → 19s

سوال ۳: تابعی ایجاد کنید که فهرستی از عدد های آمده با تاس از 1 تا 6 را بگیرد. مجموع عدد های آمده را با شرایط زیر برگردانید:

اگر 1 بیاید، بدشانسی است و تاس بعدی 0 حساب می شود.
اگر 6 بیاید، خوش شانسی است. وعدد بعدی در 2 ضرب می شود.
تعداد انداختن تاس همیشه 3 یا بیشتر خواهد بود.

نکته: حتی اگر عدد 6 بعد از 1 قرار گیرد، بدشانسی است و 6 محاسبه نمی شود، اما "اثر" 6 هنوز اتفاق می افتد.

Example:
rolls([1, 2, 3]) → 4
rolls([2, 6, 2, 5]) → 17
rolls([6, 1, 1]) → 8

سوال ۴: تابعی بنویسید که ارتفاع و عرض (m, n) و یک پارامتر اختیاری s را در ورودی بگیرد و یک آرایه با فرش n*m با استفاده از پارامتر s ایجاد کند. پارامتر s به صورت پیشفرض # است. (می توانید آن را به صورت مقدار اولیه نیز برای تابع تنظیم کنید)

Example:
makeRug(3, 5, '#') → ['#####', '#####', '#####']
makeRug(3, 5, '$') → ['$$$$$', '$$$$$', '$$$$$']
makeRug(2, 2, 'A') → ['AA', 'AA']

سوال ۵: یکی از ریاضیدانان هنگام بازی با اعداد به نکته جالبی در مورد بعضی اعداد پی برد.:

89 --> $8^1 + 9^2 = 89 * 1$
695 --> $6^2 + 9^3 + 5^4 = 1390 = 695 * 2$
46288 --> $4^3 + 6^4 + 2^5 + 8^6 + 8^7 = 2360688 = 46288 * 51$

تابعی بنویسید که یک عدد n و یک عدد p (بزرگتر مساوی 1) را به عنوان آرگومان بگیرد. عدد n همان عدد اصلی ماست. این تابع باید جمع رقم های n که به توان رسیده اند را محاسبه کند. توان ها باید از عدد p شروع شوند.

در این مثال n برابر است با 46288 و p برابر است با 3 (چون توان ها از 3 شروع شده اند):

46288 --> $4^3 + 6^4 + 2^5 + 8^6 + 8^7 = 2360688 = 46288 * 51$

همانطور که می بینید در مورد این اعداد جالب، حاصل جمع به خود عدد بخش پذیر می شود. تابع ما باید در صورت بخش پذیر بودن، مقسوم علیه را در خروجی برگرداند. یعنی در مثال بالا باید عدد 51 در خروجی برگردانده شود.

اگر حاصل جمع به عدد اولیه بخش پذیر نبود 0 برگردانده شود.

Example:
funnyNumbers(89, 1) → 1
funnyNumbers(92, 1) → 0
funnyNumbers(695, 2) → 2