High School Coding

2024 PA State Conference

This competition will involve three coding challenges of varying difficulty. We, the judges, will provide you with some information about each challenge below. You do not have to complete the challenges in the order they are listed below. You will have 2 hour to complete all of the challenges.

Challenge 1: Word Reversal

Write a function that takes in a string of one or more words, and returns the same string, but with all words that have five or more letters reversed. Strings passed in will consist of only letters and spaces. Spaces will be included only when more than one word is present.

Sample Input	Sample Output
Hey fellow warriors	Hey wollef sroirraw
This is a test	This is a test
This is another test	This is rehtona test

Challenge 2: Create Phone Number

Write a function that accepts an array of 10 integers (between 0 and 9), that returns a string of those numbers in the form of a phone number format `(###) ###-####` .

Sample Input	S ample Output
`int[]{1,2,3,4,5,6,7,8,9,0}`	(123) 456-7890
`int[]{1,1,1,1,1,1,1,1,1,1}`	(111) 111-1111

Challenge 3: Roman Numerals Encoder

Create a function taking a positive integer between 1 and 3999 (both included) as its parameter and returning a string containing the Roman Numeral representation of that integer. Modern Roman numerals are written by expressing each digit separately starting with the left most digit and skipping any digit with a value of zero. In Roman numerals 1990 is rendered: 1000=M, 900=CM, 90=XC; resulting in MCMXC. 2008 is written as 2000=MM, 8=VIII; or MMVIII. 1666 uses each Roman symbol in descending order: MDCLXVI.

For Reference:

Roman Symbol	Numeric Value
`I`	1
`V`	5

X	10
L	50
C	100
D	500
M	1,000

Sample Input	Sample Output
II	2
IV	4
D	500
MCMLIV	1954
MMVIII	2008

Challenge 4: Elemental Words

Each element in the periodic table has a symbol associated with it. For instance, the symbol for the element Yttrium is `Y` A fun thing to do is see if we can form words using symbols of elements strung together. The symbol for Einsteinium is `Es` , so the symbols for Yttrium and Einsteinium together form:

```
Y + Es = YEs
```

Yes! Ignoring capitalization, we can think of any string of letters formed by the concatenation of one or more element symbols as an *elemental word* -- per the above, `yes` is an elemental word. There is only one combination of element symbols that can form `yes` , but in general, there may be more than one combination of element symbols that can form a given elemental word. Let's call each different combination of element symbols that can form a given elemental word `word` an elemental form of `word` .

Your task is to implement the function `elementalForms(word)` , which takes one parameter (the string `word`), and returns an array (which we'll call forms). If `word` can be formed by combining element symbols from the periodic table, then forms should contain one or more sub-arrays, each consisting of strings of the form `elementName (elementSymbol)` , for each unique combination of elements that can form word.

Example

The word 'snack' can be formed by concatenating the symbols of 3 different combinations of elements:

```
_____________________________________________________
|       1        |       2        |       3        |
|----------------------------------------------------
| Sulfur    (S)  | Sulfur    (S)  | Tin       (Sn) |
| Nitrogen  (N)  | Sodium    (Na) | Actinium  (Ac) |
| Actinium  (Ac) | Carbon    (C)  | Potassium (K)  |
| Potassium (K)  | Potassium (K)  |                |
_____________________________________________________
```

So `elementalForms('snack')` should return the following array:

```
{
  {"Sulfur (S)", "Nitrogen (N)", "Actinium (Ac)", "Potassium (K)"},
  {"Sulfur (S)", "Sodium (Na)", "Carbon (C)", "Potassium (K)"},
  {"Tin (Sn)", "Actinium (Ac)", "Potassium (K)"}
}
```

For Reference:

Element Name	Symbol	Atomic Number
Calcium	Ca	20
Californium	Cf	98
Carbon	C	6
Cerium	Ce	58
Cesium	Cs	55
Chlorine	Cl	17
Chromium	Cr	24
Cobalt	Co	27
Copernicium	Cn	112
Copper	Cu	29
Curium	Cm	96
Darmstadtium	Ds	110
Dubnium	Db	105
Dysprosium	Dy	66
Einsteinium	Es	99
Erbium	Er	68
Europium	Eu	63
Fermium	Fm	100
Flerovium	Fl	114
Fluorine	F	9
Francium	Fr	87
Gadolinium	Gd	64
Gallium	Ga	31
Germanium	Ge	32
Gold	Au	79

Hafnium	Hf	72
Hassium	Hs	108
Helium	He	2
Holmium	Ho	67
Hydrogen	H	1
Indium	In	49
Iodine	I	53
Iridium	Ir	77
Iron	Fe	26
Krypton	Kr	36
Lanthanum	La	57
Lawrencium	Lr	103
Lead	Pb	82
Lithium	Li	3
Livermorium	Lv	116
Lutetium	Lu	71
Magnesium	Mg	12
Manganese	Mn	25
Meitnerium	Mt	109
Mendelevium	Md	101
Mercury	Hg	80
Molybdenum	Mo	42
Moscovium	Mc	115
Neodymium	Nd	60
Neon	Ne	10
Neptunium	Np	93
Nickel	Ni	28
Nihonium	Nh	113
Niobium	Nb	41
Nitrogen	N	7
Nobelium	No	102

Oganesson	Og	118
Osmium	Os	76
Oxygen	O	8
Palladium	Pd	46
Phosphorus	P	15
Platinum	Pt	78
Plutonium	Pu	94
Polonium	Po	84
Potassium	K	19
Praseodymium	Pr	59
Promethium	Pm	61
Protactinium	Pa	91
Radium	Ra	88
Radon	Rn	86
Rhenium	Re	75
Rhodium	Rh	45
Roentgenium	Rg	111
Rubidium	Rb	37
Ruthenium	Ru	44
Rutherfordium	Rf	104
Samarium	Sm	62
Scandium	Sc	21
Seaborgium	Sg	106
Selenium	Se	34
Silicon	Si	14
Silver	Ag	47
Sodium	Na	11
Strontium	Sr	38
Sulfur	S	16
Tantalum	Ta	73
Technetium	Tc	43

Tellurium	Te	52
Tennessine	Ts	117
Terbium	Tb	65
Thallium	Tl	81
Thorium	Th	90
Thulium	Tm	69
Tin	Sn	50
Titanium	Ti	22
Tungsten	W	74
Uranium	U	92
Vanadium	V	23
Xenon	Xe	54
Ytterbium	Yb	70
Yttrium	Y	39
Zinc	Zn	30
Zirconium	Zr	40

Sample Input	Sample Output
`Yes`	`{"Yttrium (Y)", "Einsteinium (Es)"}`
`beach`	`{"Beryllium (Be)", "Actinium (Ac)", "Hydrogen (H)"}`

`snack`	`{"Sulfur (S)", "Nitrogen (N)", "Actinium (Ac)", "Potassium (K)"},`
	`{"Sulfur (S)", "Sodium (Na)", "Carbon (C)", "Potassium (K)"},`
	`{"Tin (Sn)", "Actinium (Ac)", "Potassium (K)"}`