Algorithms

Word Combinations

You're given an array of strings *dictionary* and a single string *target*. Target string contains only characters from *a* to *z* and does not contain spaces. Please write function *find_possible_combinations* returning array of strings which represents all possible ways to form *target* from elements of *dictionary*. The order in which combinations are returned is irrelevant. Words from dictionary can be used multiple times.

Examples:

```
dictionary = ['a', 'b', 'c', 'ab', 'abc']
target = 'aabc'
find_possible_combinations(dictionary, target) => ['a abc', 'a a b c', 'a ab c']

dictionary = ['a', 'b', 'c', 'ab', 'abc']
target = 'aabcx'
find_possible_combinations(dictionary, target) => []
```

Tree Levels

Please write function *tree_levels* which takes one argument *tree* and returns its elements grouped into levels. Returned value should be an array of levels ordered from top to bottom. Each level is an array of values taken from tree nodes residing on a given tree level ordered from left to right.

Examples:

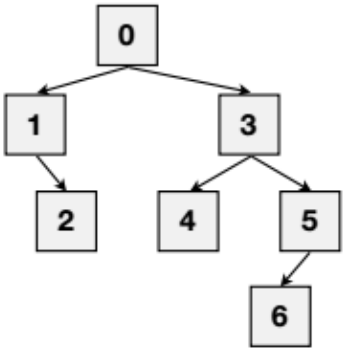
```
TreeNode = Struct.new(:value, :left, :right)

tree = TreeNode.new(0,
  TreeNode.new(1,
    nil,
    TreeNode.new(2, nil, nil)
  ),

  TreeNode.new(3,
    TreeNode.new(4, nil, nil),

    TreeNode.new(5,
      TreeNode.new(6, nil, nil),
      nil
    )
  )
)

tree_levels(tree) => [[0], [1, 3], [2, 4, 5], [6]]
```



Algorithms

Word Combina ons

You're given an array of strings dictionary and a single string target. Target string contains only characters from a to z and does not contain spaces. Please write function find_possible_combinations returning array of strings which represents all possible ways to form target from elements of dictionary. The order in which combinations are returned is irrelevant. Words from dictionary can be used multiple times.

Examples:

dictionary = ['a', 'b', 'c', 'ab', 'abc'] target = 'aabc' find_possible_combinations(dictionary, target) => ['a abc', 'a a b c', 'a ab c']

dictionary = ['a', 'b', 'c', 'ab', 'abc'] target = 'aabcx' find_possible_combinations(dictionary, target) => []

Tree Levels

Please write function tree_levels which takes one argument tree and returns its elements grouped into levels. Returned value should be an array of levels ordered from top to bottom. Each level is an array of values taken from tree nodes residing on a given tree level ordered from left to right.

Examples:

TreeNode = Struct.new(:value, :left, :right)

tree = TreeNode.new(0,

TreeNode.new(1,

nil, TreeNode.new(2, nil, nil)),

TreeNode.new(3,

TreeNode.new(4, nil, nil),

TreeNode.new(5,

TreeNode.new(6, nil, nil), nil)))

tree_levels(tree) => [[0], [1, 3], [2, 4, 5], [6]]

0

1 3

2 4 5

6

Design

Discounts

Grocery store has product inventory which consists of these three products:

- Fruit Tea (FR) - $3.11
- Strawberries (SR) - $5.00
- Coffee (CF) - $11.23

Marketing team is pushing hard to come up with pioneering offers. So far we have the following rules.

For the Fruit Tea we have buy-one-get-one-free offer. And each Strawberries item gets discounted down to $4.50 if one buys 3 or more of those. These rules are independent from each other and can both apply to the same checkout.

We expect that marketing team would change their minds pretty often and there can be more rules like that. So we would like to anticipate that.

Please implement checkout system which fulfills the above requirements and provides interface similar to the following:

```
co = Checkout.new(rule1, rule2, ...)
co.scan('FR')
co.scan('CF')
...
total_cost = co.total
```

Examples:

```
Items: FR, SR, FR, FR, CF
Total: $22.45

Items: FR, FR
Total: $3.11

Items: SR, SR, FR, SR
Total: $16.61
```

Design

Discounts

Grocery store has product inventory which consists of these three products:

• Fruit Tea (FR) - $3.11

• Strawberries (SR) - $5.00

• Coffee (CF) - $11.23

Marketing team is pushing hard to come up with pioneering offers. So far we have the following rules.

For the Fruit Tea we have buy-one-get-one-free offer. And each Strawberries item gets discounted down to $4.50 if one buys 3 or more of those. These rules are independent from each other and can both apply to the same checkout.

We expect that marketing team would change their minds pretty often and there can be more rules like that. So we would like to anticipate that.

Please implement checkout system which fulfills the above requirements and provides interface similar to the following:

co = Checkout.new(rule1, rule2, ...) co.scan(‘FR’) co.scan(‘CF’) ... total_cost = co.total

Examples:

Items: FR, SR, FR, FR, CF Total: $22.45

Items: FR, FR Total: $3.11

Items: SR, SR, FR, SR Total: $16.61

Web UI

Parcel Delivery

The MoveFast company provides parcel delivery services. Their typical customers buy stuff on eBay asking this stuff to be delivered (usually for free) to one of MoveFast warehouses. MoveFast accumulates goods, then repacks and sends those to customers for small affordable fee. In order to provide best user experience MoveFast company would like to have online parcel delivery calculator on their website.

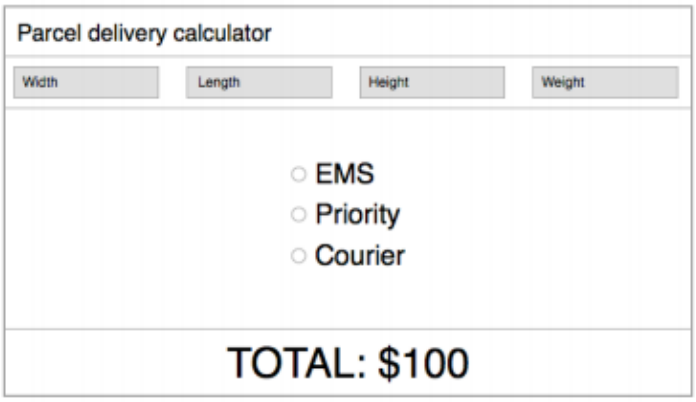
A parcel has four attributes: *Weight*, *Length*, *Height* and *Width*.

It can be delivered via one of three carriers: *EMS*, *Priority* or *Courier*. Each carrier implies its own costs and limits:

- Processing fee (USD)
- Price per unit of weight (USD/kg)
- Max weight (kg)

MoveFast processing fee is fixed at $15 per parcel.

Please implement online calculator according to the following mock:



Additional requirements:

- Price calculation should be instant with no server communication.
- Should parcel attributes exceed carrier's processable weight limit, that carrier option should be disabled
- If all carriers are disabled, TOTAL section should display error message "This parcel is over processing limits and could not be delivered."
- TOTAL price should be calculated as: MoveFast fee + Carrier's Processing fee + weight * Carrier's Price per 1 kg.

Sample carrier attributes:

- **EMS**. Processing fee: $10; Price per kg: $1.5; Max Processable Weight: 30kg
- **Priority**. Processing fee: $15; Price per kg: $5; Max Processable Weight: 22kg
- **Courier**. Processing fee: $20; Price per kg: $15; Max Processable Weight: 10kg

Web UI

Parcel Delivery

The MoveFast company provides parcel delivery services. Their typical customers buy stuff on eBay asking this stuff to be delivered (usually for free) to one of MoveFast warehouses. MoveFast accumulates goods, then repacks and sends those to customers for small affordable fee. In order to provide best user experience MoveFast company would like to have online parcel delivery calculator on their website.

A parcel has four attributes: Weight, Length, Height and Width.

It can be delivered via one of three carriers: EMS, Priority or Courier. Each carrier implies its own costs and limits:

• Processing fee (USD)

• Price per unit of weight (USD/kg)

• Max weight (kg)

MoveFast processing fee is fixed at $15 per parcel.

Please implement online calculator according to the following mock:

Additional requirements:

• Price calculation should be instant with no server communication.

• Should parcel attributes exceed carrier's processable weight limit, that carrier option should be disabled

• If all carriers are disabled, TOTAL section should display error message "This parcel is over processing limits and could not be delivered."

• TOTAL price should be calculated as: MoveFast fee + Carrier's Processing fee + weight * Carrier's Price per 1 kg.

Sample carrier attributes:

• EMS. Processing fee: $10; Price per kg: $1.5; Max Processable Weight: 30kg

• Priority. Processing fee: $15; Price per kg: $5; Max Processable Weight: 22kg

• Courier. Processing fee: $20; Price per kg: $15; Max Processable Weight: 10kg