**Work Log for SageTeck**

By: Bilal Ahmad

10-08-2024

## Task 1: Returning Missing Alphabet

### Initial approach:

Assuming the input array can only have either all upper case or all lower-case alphabets I declared two arrays separately for all upper-case alphabets and all lower-case alphabets. Then I checked the if first character of the input array was upper case or lower case and accordingly stored lower case or upper caser arrays in the referenced alphabet. Then loop till the size of the input array and place a check for character mismatch with the referenced alphabet array and return that.

### Challenges Faced:

At first I thought using ASCII values for the corresponding characters in the past characters array and incrementing them would have been a good idea but it given the high level language it wasn’t that easy relative to the approach I decided to implement.

### Pseudo Code:

FUNCTION missingLetter(charArray):

DECLARE uppercaseAlphabet AS "ABCDEFGHIJKLMNOPQRSTUVWXYZ".split('')

DECLARE lowercaseAlphabet AS "abcdefghijklmnopqrstuvwxyz".split('')

SET firstChar TO charArray[0]

DECLARE referenceAlphabet

IF firstChar is uppercase:

SET referenceAlphabet TO uppercaseAlphabet

ELSE:

SET referenceAlphabet TO lowercaseAlphabet

SET startIndex TO index of firstChar in referenceAlphabet

FOR i FROM 0 TO length of charArray - 1:

IF charArray[i] is NOT EQUAL to referenceAlphabet[startIndex + i]:

RETURN referenceAlphabet[startIndex + i

RETURN null

function missingLetter(charArray):

firstChar = charArray[0]

isUppercase = (firstChar is equal to firstChar.toUpperCase())

startIndex = ASCII value of firstChar

for i from 0 to length of charArray - 1:

expectedChar = Character corresponding to (startIndex + i)

if charArray[i] is not equal to expectedChar:

return expectedChar

return null

**Time complexity:** O(n)  
**Space Complexity:** O(1)

### Improved approach:

**1. Avoid Splitting the Alphabet Strings:**

The split('') operation on the alphabet strings can be replaced by working directly with the strings since indexing into a string is as efficient as indexing into an array.

**2. Simplify Reference Alphabet Selection:**

Instead of using two separate strings for uppercase and lowercase alphabets, we can use a single string and adjust the starting point based on whether the first character is uppercase or lowercase.

**3. Avoid indexOf in the Alphabet:**

The indexOf method scans the string to find the starting index, which can be avoided by directly calculating the index using ASCII values. This makes the operation constant time without scanning.

The complexity remains the same but in general the approach is improved resulting in more better approach efficient approach.  
  
Pseudo Code:

function missingLetter(charArray):

firstChar = charArray[0]

isUppercase = (firstChar is equal to firstChar.toUpperCase())

startIndex = ASCII value of firstChar

for i from 0 to length of charArray - 1:

expectedChar = Character corresponding to (startIndex + i)

if charArray[i] is not equal to expectedChar:

return expectedChar

return null

### Assumption 2:

Assuming the input array can have both upper- and lower-case characters we can follow the same approach but have on one reference array and convert each character of input array to lowercase and check as we checked before.

## Task 2: Return Prices

### Initial Approach:

First start a loop to parse each grocery item and then inside each grocery item parse it again until first number is detected and then that digit is parsed until the number is not detected or a parenthesis is detected.

### Challenges Faced:

My first thought process was to detect the $ symbol in side the passed array and then iterate from the next index, but the concern here was, what if the currency symbol is changed so to make it more dynamic and widely usable I went with detecting the first number as grocery item names don’t usually have numbers in them.

### Pseudo Code:

FUNCTION getPrices(list):

CREATE an empty array prices

FOR each item in list:

SET charprice to an empty string

FOR each character in item:

IF character is a digit (between '0' and '9'):

SET index k to the current character's position

WHILE k is within item and the character at position k is a digit or a decimal point:

ADD character at position k to charprice

INCREMENT k

ADD parseFloat(charprice).toFixed(2) to prices array

BREAK inner loop

RETURN prices array

**Time Complexity:** O(n\*m)

**Space Complexity:** O(n)

## Task 3: Checking Comments

### Initial approach:

The approach was to first check the first character of the pattern if it is a ‘\*’ the pattern is directly a miss match if it is a ‘/’ then check for the next character if it is not the end of the array and not a ‘/’ then skip the next character and continue, if it is a ‘\*’ and not the end of array then check the third character after that for end of array and another ‘/’ and the second character for ‘\*’ to skip the next 3 characters (skip ‘/\*\*/’) then continue. If the array does not fulfill these conditions, then it returns false.

### Challenges Faced:

The only challenge I faced in this was what if there were odd number of characters in the past array and the last character was a ‘/’ for that I just checked the index with the size of the array to not let that happen.

Pseudo Code:  
FUNCTION commentsCorrect(pattern):

SET i to 0

WHILE i < length of pattern:

IF pattern[i] is '/':

IF i + 1 < length of pattern AND pattern[i + 1] is '/':

INCREMENT i by 1 (skip the next character)

ELSE IF i + 1 < length of pattern AND pattern[i + 1] is '\*':

IF i + 3 < length of pattern AND pattern[i + 2] is '\*' AND pattern[i + 3] is '/':

INCREMENT i by 3 (skip the next three characters)

ELSE:

RETURN false

ELSE:

RETURN false

ELSE IF pattern[i] is '\*':

RETURN false

INCREMENT i by 1

RETURN true

**Time Complexity:** O(n)

**Space Complexity:** O(1)

## Task 4: Error Correction

The $agentexist variable is reset to false inside the loop that iterates over the security groups, which is a logical error in the code. This means that $agentexist will be set wrongly to false if the current security group does not have the "AGENTE" role and a prior security group did. This can be fixed by setting $agentexist to true as soon as "AGENTE" is located, and not resetting it to false later.

Corrected Code:

global $current\_user

$roles1 = ACLRole::getUserRoleNames($current\_user->id);

$agentexist2 = false;

foreach ($roles1 as $newrole1) {

if ($newrole1 == 'AGENTE') {

$agentexist2 = true;

break;

}

}

// Get the current user's ID

$current\_user\_id = $current\_user->id;

// Get the security groups of the current user

$security\_groups = SecurityGroup::getUserSecurityGroups($current\_user\_id);

$agentexist = false; // Initialize outside the loop

foreach ($security\_groups as $sg\_id => $sg\_name) {

$security\_group\_bean = BeanFactory::getBean('SecurityGroups', $sg\_id);

if ($security\_group\_bean->load\_relationship('aclroles')) {

$test = $security\_group\_bean->aclroles->getBeans();

foreach ($test as $newrole) {

if ($newrole->name == 'AGENTE') {

$agentexist = true;

break;

}

}

}

}

if ($agentexist2 || $agentexist) {

echo "success";

}