Assignment #1

COEN 278 - Fall 2023

Question 1

Redefine the following method of Array class:

a. Array#select

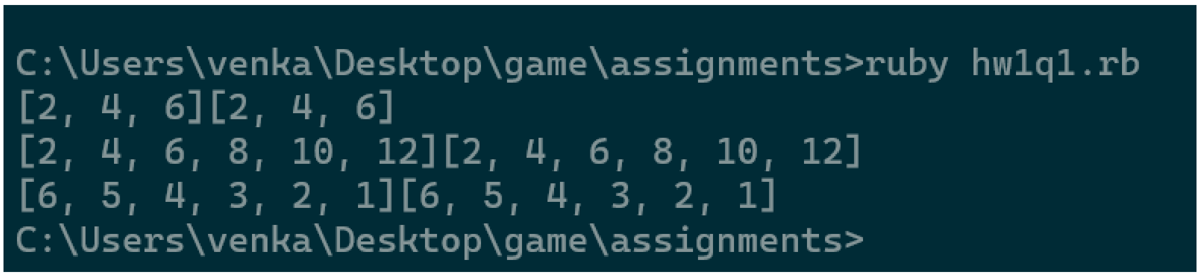
b. Array#map

c. Array#reverse

Answer

```ruby
class Array
    # redefine to exactly how select method works
    def my_select
        result = []
        # << is used for appending item onto a list
        each { |item| result << item if yield(item) }
        result
    end
    # redefine to exactly how map method work
    def my_map
        result = []
        each {|item| result << yield(item)}
        result
    end
    # redefine to exactly how reverse method work
    def my_reverse
        result = []
        (size - 1).downto(0) { |i| result << self[i] }
        result
    end
end


nums = [1,2,3,4,5,6]
even = nums.my_select{|x| x.even?}
double = nums.my_map{|x| 2*x}
reverse = nums.my_reverse()
original_even = nums.select{|x| x.even?}
original_double = nums.map{|x| 2*x}
original_reverse = nums.reverse()
print(even,original_even,"\n")
print(double,original_double,"\n")
print(reverse,original_reverse)
```

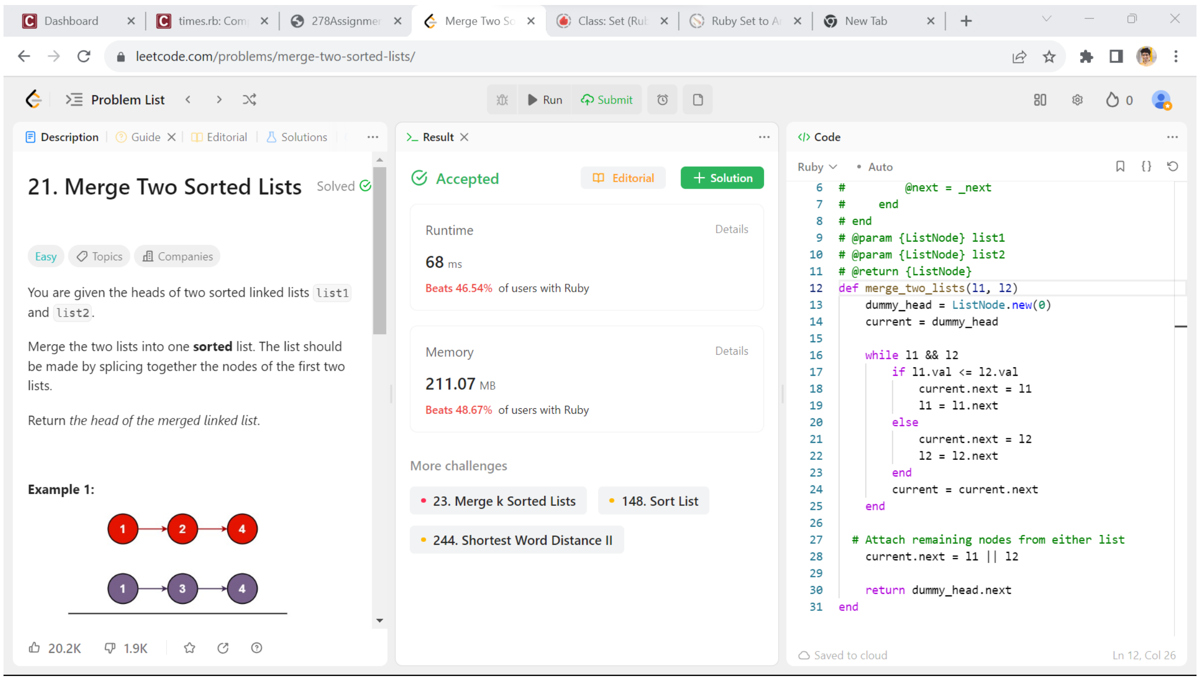
The following showcases the output and is checked against the original inbuilt methods and my methods, which do the same thing.



```
C:\Users\venka\Desktop\game\assignments>ruby hw1q1.rb
[2, 4, 6][2, 4, 6]
[2, 4, 6, 8, 10, 12][2, 4, 6, 8, 10, 12]
[6, 5, 4, 3, 2, 1][6, 5, 4, 3, 2, 1]
C:\Users\venka\Desktop\game\assignments>
```

Question 2 Solve the following leetcode problem using Ruby language 21. Merge Two Sorted Lists (submit screen shot of submission and source code file)

Answer



21. Merge Two Sorted Lists Solved

You are given the heads of two sorted linked lists `list1` and `list2`.

Merge the two lists into one **sorted** list. The list should be made by splicing together the nodes of the first two lists.

Return *the head of the merged linked list.*

Example 1:

Accepted

Runtime
68 ms
Beats 46.54% of users with Ruby

Memory
211.07 MB
Beats 48.67% of users with Ruby

23. Merge k Sorted Lists
148. Sort List
244. Shortest Word Distance II

```ruby
#         @next = _next
#     end
# end
# @param {ListNode} list1
# @param {ListNode} list2
# @return {ListNode}
def merge_two_lists(l1, l2)
    dummy_head = ListNode.new(0)
    current = dummy_head

    while l1 && l2
        if l1.val <= l2.val
            current.next = l1
            l1 = l1.next
        else
            current.next = l2
            l2 = l2.next
        end
        current = current.next
    end

  # Attach remaining nodes from either list
    current.next = l1 || l2

    return dummy_head.next
end
```

Saved to cloud
Ln 12, Col 26

the source code for all the problems will be attached individually

```ruby
# Definition for singly-linked list.
# class ListNode
#     attr_accessor :val, :next
#     def initialize(val = 0, _next = nil)
#         @val = val
#         @next = _next
#     end
# end
# @param {ListNode} list1
# @param {ListNode} list2
# @return {ListNode}
def merge_two_lists(l1, l2)
    dummy_head = ListNode.new(0)
    current = dummy_head

    while l1 && l2
        if l1.val <= l2.val
            current.next = l1
            l1 = l1.next
        else
            current.next = l2
            l2 = l2.next
        end
        current = current.next
    end

  # Attach remaining nodes from either list
    current.next = l1 || l2

    return dummy_head.next
end
```

Question 3

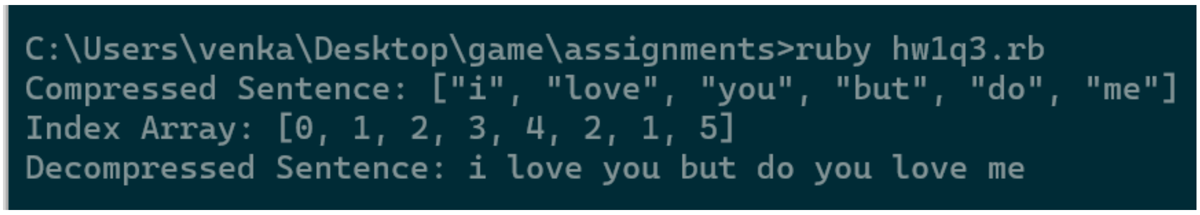
Write a class for compressing a sentence. By compress a sentence, we mean to remove duplicate words. For example, a sentence "i love you but do you love me" will be compressed into "i love you but do me". When you create an object of this class, you pass a sentence argument (a string), then the object is initialized with the compressed result of this string as the attribute of the object. The compressed result will be saved in an array of each word of the initial sentence. You also need another array to remember index of word in compressed array for decompress purpose. For example: assuming the name of your class is Compress, to create an object: obj = Compress.new(“i love you but do you love me”) #there are duplicate words in it then there will be two attributes created inside the object to hold two values: [“i”, “love”, “you”, “but”, “do”, “me”] # duplicate word removed (compressed) [0, 1, 2, 3, 4, 2, 1, 5] # index to the original array to represent original sentence You task: add an instance method to return the original string (not compressed)

Answer

```
class Compress
    attr_accessor :compressed_sentence, :index_array

    def initialize(sentence)
        @compressed_sentence = []
        @index_mapping = {}
        words = sentence.split
        compressed_index = 0

        words.each do |word|
            unless @index_mapping.key?(word)
                @compressed_sentence << word
                @index_mapping[word] = compressed_index
                compressed_index += 1
            end
        end

        @index_array = words.map { |word| @index_mapping[word] }
    end

    def decompress
        original_sentence = @index_array.map { |index|
@compressed_sentence[index] }
        original_sentence.join(' ')
    end
end
```

```
#example
sentence = "i love you but do you love me"
compressor = Compress.new(sentence)
puts "Compressed Sentence: #{compressor.compressed_sentence}"
puts "Index Array: #{compressor.index_array}"
puts "Decompressed Sentence: #{compressor.decompress}"
```

Running the above file gives out the following result:



```
C:\Users\venka\Desktop\game\assignments>ruby hw1q3.rb
Compressed Sentence: ["i", "love", "you", "but", "do", "me"]
Index Array: [0, 1, 2, 3, 4, 2, 1, 5]
Decompressed Sentence: i love you but do you love me
```