For each of the student submissions below, what feedback would you give to help them improve their submission, with a goal of eventually emulating the instructor solution (provided below)? Assume that the student will be able to resubmit their assignment after reviewing this feedback.

Instructor solution:

```ruby
def combine_anagrams(words)
    words.group_by { |word| word.downcase.chars.sort }.values
end
```

Please read all the submissions before beginning to give feedback. Please don't move on to the next page until you have completed all the questions on this page.

Submission 1

```ruby
def combine_anagrams(words)
  sortedchars = []
  words.each { |word| (sortedchars << word.downcase.split(//).sort) }
  h = {}
  sortedchars.uniq.each { |chars| h[chars] = [] }
  words.each { |word| (h[word.downcase.split(//).sort] << word) }
  h.values
end
```

(For our reference: 18.8 224)

*

A simpler way to get the characters of a string is by using .chars. Consider looking through the Enumerable docs for a better-suited function.

Submission 2

```ruby
def combine_anagrams(words)
  a = Array.new
  b = Array.new
  words.each do |x|
    a = words.select do |y|
      (y.downcase.chars.sort.join == x.downcase.chars.sort.join)
    end
    b.push(a) unless (a == [])
    words = (words - a)
  end
  return b
end
```

(For our reference: 24.2 250)

*

Try to think of a solution that doesn't involve writing a double loop over all of the elements, look through the Ruby's documentation and see if you can find a way to group certain elements together. A couple other comments: there are simpler ways of creating an array (a = []) for

Submission 3

```ruby
def combine_anagrams(words)
  anagrams = words.group_by { |word| word.downcase.chars.sort }.values
end
```

(For our reference: 6.9 348)

*

Almost, think about what is returned when you assign something to a variable.

Submission 4

```ruby
def combine_anagrams(words)
  combined_list = []
  words.each do |word|
    sorted = word.chars.sort_by(&:downcase).join
    inserted = false
    combined_list.each do |anagram|
      compare1 = sorted.downcase
      compare2 = anagram[0].chars.sort_by(&:downcase).join.downcase
      puts(((compare1 + " ") + compare2))
      puts(" ")
      if (compare1 == compare2) then
        (anagram << word)
        inserted = true
      end
    end
    (combined_list << [word]) if (inserted == false)
  end
  return combined_list
end
```

(For our reference: 38.6 352)

*

Try to think of a simpler algorithm to solve this. Reading through the available functions in the Enumerable docs may give you a better sense on how to group certain elements together. Also, for this problem you don't have to print anything.

Submission 5

```ruby
def combine_anagrams(words)
  result = words.group_by { |word| word.downcase.chars.sort }.values
  return result
end
```

(For our reference: 6.9 672)

*

So close! Think about what Ruby will return if you don't put a "return" statement in.

Submission 6

```ruby
def combine_anagrams(words)
  hash = {}
  words.each do |word|
    sort_word = word.downcase.chars.sort.join
    hash[sort_word] ||= []
    (hash[sort_word] << word)
  end
  return hash.values
end
```

(For our reference: 10.9 610)

*

Try to do this without creating an empty hash. You should look through the Enumerable docs to see if there is a more-equipped function.

* Indicates Response Required

For each of the student submissions, rate the autogenerated hints on a scale of 1 to 5.

5 - Highly relevant: covers a key concept that should be improved.
4 - Somewhat relevant: helpful in a lesser way.
3 - Neutral: not helpful.
2 - Irrelevant: this hint is irrelevant
1 - Harmful: this hint is completely wrong and harmful

Please also comment on the degree to which these hints capture the feedback you gave in the previous step.

Submission 1

```
1 def combine_anagrams(words)
2   sortedchars = []
3   words.each { |word| (sortedchars << word.downcase.split(//).sort) }
4   h = {}
5   sortedchars.uniq.each { |chars| h[chars] = [] }
6   words.each { |word| (h[word.downcase.split(//).sort] << word) }
7   h.values
8 end
```

(For our reference: 18.8 224)

Here's the feedback you gave for this submission:

A simpler way to get the characters of a string is by using .chars. Consider looking through the Enumerable docs for a better-suited function.

*** Here's the feedback the autograder gave:**

To improve your style, consider...

...using a call to new.	★★☆☆☆
...using a call to join.	★★★☆☆
...using a method that produces strings.	★★★★☆
...not using a call to uniq.	★★★★★

*** Comment here on the degree to which degree these hints capture the feedback you gave this submission.**

Some of the comments seem to not match with the instructor solution, but not using a call to uniq is a good idea. The method that

Submission 2

```
1 def combine_anagrams(words)
2   a = Array.new
3   b = Array.new
4   words.each do |x|
5     a = words.select do |y|
```

```
 6       (y.downcase.chars.sort.join == x.downcase.chars.sort.join)
 7     end
 8     b.push(a) unless (a == [])
 9     words = (words - a)
10   end
11   return b
12 end
```

(For our reference: 24.2 250)

Here's the feedback you gave for this submission:

Try to think of a solution that doesn't involve writing a double loop over all of the elements, look through the Ruby's documentation and see if you can find a way to group certain elements together. A couple other comments: there are simpler ways of creating an array (a = []) for example and line 9 will cause an error. You cannot remove an element from an array by subtraction. Furthermore, it is generally bad practice to modify an array that you are iterating over, especially if you might modify its size.

*** Here's the feedback the autograder gave:**

To improve your style, consider...

...using a call to uniq.	★☆☆☆☆
...not using a method that produces hashes.	★★☆☆☆
...not using a method that produces strings.	★★☆☆☆
...not using a call to select.	★★☆☆☆

*** Comment here on the degree to which degree these hints capture the feedback you gave this submission.**

I don't think these comments captured really any part of the problems here.

Submission 3

```
1 def combine_anagrams(words)
2   anagrams = words.group_by { |word| word.downcase.chars.sort }.values
3 end
```

(For our reference: 6.9 348)

Here's the feedback you gave for this submission:

Almost, think about what is returned when you assign something to a variable.

*** Here's the feedback the autograder gave:**

To improve your style, consider...

No feedback generated	★★★★★

*** Comment here on the degree to which degree these hints capture the feedback you gave this submission.**

I gave more hints in my feedback, but a student should be able to figure this out on their own, so no feedback is OK, especially

Submission 4

```
1 def combine_anagrams(words)
2   combined_list = []
3   words.each do |word|
4     sorted = word.chars.sort_by(&:downcase).join
5     inserted = false
6     combined_list.each do |anagram|
7       compare1 = sorted.downcase
8       compare2 = anagram[0].chars.sort_by(&:downcase).join.downcase
9       puts(((compare1 + " ") + compare2))
10      puts(" ")
11      if (compare1 == compare2) then
12        (anagram << word)
13        inserted = true
14      end
15    end
16    (combined_list << [word]) if (inserted == false)
17  end
18  return combined_list
19 end
```

(For our reference: 38.6 352)

Here's the feedback you gave for this submission:

Try to think of a simpler algorithm to solve this. Reading through the available functions in the Enumerable docs may give you a better sense on how to group certain elements together. Also, for this problem you don't have to print anything.

*** Here's the feedback the autograder gave:**

To improve your style, consider...

...using a call to sort.	★★★★★
...using a call to push.	★★☆☆☆
...using a call to split.	★★★★★
...not using a call to chars.	★☆☆☆☆
...not using a call to puts.	★★★★★
...not using a call to sort_by.	★★★★★

*** Comment here on the degree to which degree these hints capture the feedback you gave this submission.**

I think this gives a good first step towards a solution.

Submission 5

```ruby
def combine_anagrams(words)
  result = words.group_by { |word| word.downcase.chars.sort }.values
  return result
end
```

(For our reference: 6.9 672)

Here's the feedback you gave for this submission:

So close! Think about what Ruby will return if you don't put a "return" statement in.

*** Here's the feedback the autograder gave:**

To improve your style, consider...

No feedback generated 

*** Comment here on the degree to which degree these hints capture the feedback you gave this submission.**

It would be helpful to remind the student that they don't have to explicitly say return

Submission 6

```ruby
def combine_anagrams(words)
  hash = {}
  words.each do |word|
    sort_word = word.downcase.chars.sort.join
    hash[sort_word] ||= []
    (hash[sort_word] << word)
  end
  return hash.values
end
```

(For our reference: 10.9 610)

Here's the feedback you gave for this submission:

Try to do this without creating an empty hash. You should look through the Enumerable docs to see if there is a more-equipped function.

*** Here's the feedback the autograder gave:**

To improve your style, consider...

...using a call to split. 

...not using a call to chars. 

*** Comment here on the degree to which degree these hints capture the feedback you gave this submission.**

I think the best comment would be something about grouping here, since this is a pretty close solution.

* Indicates Response Required

Close