1. Write a function called `frequency_string(chars)`, which given an array of single-character strings, returns one string that includes each unique character from the input array followed by its frequency.

   For example:

   ```
   frequency_string(%w{a c # 2 a a #}) => 'a3c1#221'
   ```

   Note: the ordering of character/count pairs in the return value string doesn't matter.

   Solution:

   ```ruby
   def frequency_string(chars)
       frequencies = {}
       chars.each do |char|
           if frequencies.has_key? char
               frequencies[char] += 1
           else
               frequencies[char] = 1
           end
       end
       results = ""
       frequencies.each do |char,count|
           results << char << count.to_s
       end
       results
   end

   # or

   def frequency_string(chars)
       frequencies = Hash.new(0)
       chars.each do |char|
           frequencies[char] += 1
       end
       return frequencies.flatten.join
   end
   ```

2. Rewrite the following lines of code to minimize the number of characters used while keeping the same behavior.

   ```ruby
   if (!some_boolean)
       puts 'hello world'
   end
   ```

   Solution:

   ```ruby
   puts 'hello world' unless some_boolean
   ```

3. Given the following method definition, which the method invocations are not legal?

```ruby
def do_something(mode, *args, options)
    if mode == :debug
        args.each do |arg|
            puts arg
        end
    end
end
```

(a) `do_something(:debug, ["pumpkin", "peach", "porridge"], { :print => false, :make_soup => true })`

(b) `do_something(:debug, sum 4, 2, 3, 8, {})`

(c) `do_something :development, "/etc/passwd", "trololololol", :hello => ['konichiwa', 'ni hao', 'bonjour', 'hola'], :conditions => :all`

(d) `args= [ ["beach","forest","city"],{} ] do_something :test, *args`

(e) `do_something :prod, {:hello=>:world}, :i_like => 'ice cream'`

> **Solution:** Only the second one is illegal. The call to the sum method is ambiguous: what are the arguments to sum, and what are the args to do_something?

4. Write a regular expression to extract the first and last names from a Moodle-generated homework file name. Moodle uses a pretty ugly syntax for its file names, including the first name, an optional preferred name in parentheses, the last name and some text that indicates the ID of the file submission and some other information. Here are two examples:

   - `Robert (Bob) Metcalfe_585295_assignsubmission_onlinetext_onlinetext.html`
   - `Robert Metcalfe_585295_assignsubmission_onlinetext_onlinetext.html`

   Your regex should result in the special variable `$1` storing the first name and the special variable `$2` storing the last name. You should ignore the preferred name.

> **Solution:** Here's one possible solution:
>
> `/^([a-z]+)\s+(\([a-z]+\)\s+)?([a-z]+)_/i`