Regular expressions aren't just limited to searching with characters and strings, the real power of regular expressions come from using **metacharacters**. Remember that metacharacters are characters that can be used to represent other characters. To take full advantage of all of the metacharacters we should use grep's cousin egrep, which just extends grep's capabilities. The first metacharacter we should discuss is the "." (period) metacharacter, which represents *any* character. If for example I wanted to search states.txt for the character "i", followed by any character, followed by the character "g" I could do so with the following command:

```
egrep "i.g" states.txt

## Virginia
## Washington
## West Virginia
## Wyoming
```

The regular expression "i.g" matches the sub-string "irg" in V*irg*inia, and West V*irg*inia, and it matches the sub-string "ing" in Wash*ing*ton and Wyom*ing*. The period metacharacter is a stand-in for the "r" in "irg" and the "n" in "ing" in the example above. The period metacharacter is extremely liberal, for example the command egrep "." states.txtwould return every line of states.txt since the regular expression "." would match one occurrence of any character on every line (there's at least one character on every line).

Besides characters that can represent other characters, there are also metacharacters called **quantifiers** which allow you to specify the number of times a particular regular expression should appear in a string. One of the most basic quantifiers is "+" (plus) which represents one or more occurrences of the preceeding expression. For example the regular expression "s+as" means: one or more "s" followed by "as". Let's see if any of the state names match this expression:

```
egrep "s+as" states.txt

## Arkansas
## Kansas
```

Both Arkansas and Kansas match the regular expression "s+as". Besides the plus metacharacter there's also the "*" (star) metacharacter which represents zero or more occurrences of the preceding expression. Let's see what happens if we change "s+as" to "s*as":

```
egrep "s*as" states.txt

## Alaska
## Arkansas
## Kansas
## Massachusetts
## Nebraska
## Texas
## Washington
```

As you can see the star metacharacter is much more liberal with respect to matching since many more state names are matched by "s*as". There are more specific quantifies you can use beyond “zero or more” or “one or more” occurrences of an expression. You can use curly brackets ({ }) to specify an exact number of occurrences of an expression. For example the regular expression "s{2}" specifies exactly two occurrences of the character “s”. Let’s try using this regular expression:

```
egrep "s{2}" states.txt

## Massachusetts
## Mississippi
## Missouri
## Tennessee

```

Take note that the regular expression "s{2}" is equivalent to the regular expression "ss". We could also search for state names that have between two and three adjacent occurrences of the letter “s” with the regular expression "s{2,3}":

```
egrep "s{2,3}" states.txt

## Massachusetts
## Mississippi
## Missouri
## Tennessee

```

Of course the results are the same because there aren’t any states that have “s” repeated three times.

You can use a capturing group in order to search for multiple occurrences of a string. You can create capturing groups within regular expressions by using parentheses ("( )"). For example if I wanted to search states.txt for the string “iss” occurring twice in a state name I could use a capturing group and a quantifier like so:

```
egrep "(iss){2}" states.txt

## Mississippi

```

We could combine more quantifiers and capturing groups to dream up even more complicated regular expressions. For example, the following regular expression describes three occurrences of an “i” followed by two of any character:

```
egrep "(i.{2}){3}" states.txt

## Mississippi

```

The complex regular expression above still only matches “Mississippi”.

Mark as completed