Substitution

`$0`	entire substring matching the pattern
`$1`	substring matched by the part of the pattern beginning with the first (and ending with the matching)
…	…
`$9`	

there are variations in syntax from one application to the next, especially beyond the basic core

Give a search pattern and a replace pattern that could be used to convert seven-digit telephone numbers in the format `xxx-xxx-xxxx` to the format `(xxx)xxx-xxxx`.

Answer: The search pattern is `([0-9]{3})-([0-9]{3}-[0-9]{4})` and the replace pattern is `($1)$2`

Discussion: The entire string matching the search pattern is replaced by the replace pattern. Use `()` to denote groups of matched elements that should appear in the result, and use `$1`, `$2`, etc to refer to those matched groups.

Back References

`\0`	entire substring matching the pattern
`\1`	substring matched by the part of the pattern beginning with the first (and ending with the matching)
…	…
`\9`	

there are variations in syntax from one application to the next, especially beyond the basic core

Write a pattern that matches all strings in the language $L = \{a^n b a^n \mid n \geq 0\}$.

Answer: `(a*)b\1`

Discussion: `(a*)b(a*)` matches all strings containing exactly one b, but it doesn't require that the number of as before and after the b be the same. The backreference `\1`, which refers to what was matched by the first set of parens, makes it possible to specify that the substring following the b must be the same as whatever substring preceeded the b.

note that including backreferences mean that the search pattern is no longer a regular expression – it can match non-regular languages – even though it is commonly referred to as such