sed Examples:

All of these examples will operate on a file named test.dat that contains the following lines of text:

```
The Birthday Song

Happy Birthday to You
Happy Birthday to You
Happy Birthday Dear (name)
Happy Birthday to You.
```

Example 1: Output the lines that contain the string Happy

Command line: sed '/Happy/p' test.dat

Output:

```
The Birthday Song

Happy Birthday to You
Happy Birthday to You
Happy Birthday to You
Happy Birthday to You
Happy Birthday Dear (name)
Happy Birthday Dear (name)
Happy Birthday to You.
Happy Birthday to You.
```

Explanation:

sed '/Happy/p' test.dat
The string to search for is delimited by / /, in this case Happy: /Happy/
The **p** following /Happy/**p** says to print out all of the lines that match the search string. By default, sed will also print all of the lines of input, so in this example any line that contains the string /Happy/ will be printed twice.

To print **only** the lines in the file that contain Happy, use the –n option:

Command line: sed –n '/Happy/p' test.dat
Output:

```
Happy Birthday to You
Happy Birthday to You
Happy Birthday Dear (name)
Happy Birthday to You.
```

Example 2: Change the string **(name)** with the name Samantha

Command line: sed 's/(name)/Samantha/' test.dat

Output:

The Birthday Song

Happy Birthday to You
Happy Birthday to You
Happy Birthday Dear Samantha
Happy Birthday to You.

Explanation:

sed 's/(name)/Samantha/' test.dat

This sed command uses the substitute command **s**.
The format of the **s** command is to /search for this string/replace it with this one/
and this examples searches for the string (name) and replaces it with Samantha.

The output of this sed command is a copy of every input line with the substitution edits applied. If an input line does not match the string being replaced it will be printed exactly as it is was read in. If the input line does have a match for the search string the line will be printed after the edit (substitution) has been made.

The command sed -n 's/(name)/Samantha/' test.dat
will suppress all output because of the –n option.

To print only the lines where a substitution was applied use the **p** command following the substitution: sed -n 's/(name)/Samantha/p' test.dat

Output: Happy Birthday Dear Samantha

Example 3: Change the string **(name)** with the name Samantha, and change the order of the words so it read: Dear Samantha, Happy Birthday

Command line: sed 's/\(.*\)Dear (name)/Dear Samantha, \1/' test.dat

Output:

The Birthday Song

Happy Birthday to You
Happy Birthday to You
Dear Samantha, Happy Birthday
Happy Birthday to You.

Explanation:

sed 's/\(.*\)Dear (name)/Dear Samantha, \1/' test.dat

The input line that is changed is : Happy Birthday Dear (name)
The sed command will store the beginning of the line – up to Dear (name) in a variable named \1. This is done by telling sed to match zero or more characters .* and by telling it to store the matching in string in \1 by using \(and .\)
so the \(.*\) stores the string. If you use more than one pair of \(\) to capture string matches, they are stored in variables \2..\9.

The sed command is using the substitute command, the string that is being matched is /\(.*\)Dear (name)/ which is some characters before Dear (Name)

The string it is being replaced by is the string Dear Samantha, \1
which means replace it with the two words Dear Samantha followed by a comma , followed by the string that was captured and stored in \1.

There other ways to do this – but this example shows how to capture parts of a string.