C03-2 Extracting substrings

Extracting or replacing parts of a substring is quite straight forward but requires some more typing than e.g. in Python. The main function you will use is the `substring` function.

Let's extract parts of a given string starting with the second and ending with the seventh character.

```
s <- c("Hello World")
substring(s, 2, 7)
```

```
## [1] "ello W"
```

The second argument in the substring function defines the starting, the third the ending position of the string extraction.

If you apply the function to a vector of characters, it is applied to each entry (surprise, surprise).

```
v <- c("Hello Earth", "Hi Pluto")
substring(v, 2, 7)
```

```
## [1] "ello E" "i Plut"
```

A more genreal task might be to extract the second part of a character string which is of type " " i.e. with the two strings separated by blank or any other specific character. In the example above, this would imply the following:

```
substring(v[1], 7, 11)
```

```
## [1] "Earth"
```

```
substring(v[2], 4, 8)
```

```
## [1] "Pluto"
```

To automatize that, one could use the `regexpr` function which looks for a pattern in a character element and returns the index of this.

```
regexpr(" ", v[1])
```

```
## [1] 6
## attr(,"match.length")
## [1] 1
## attr(,"useBytes")
## [1] TRUE
```

The first line gives the position within the character string. The other two lines are just attributes with additional information. Of course, this can also be applied to an entire vector:

```
regexpr(" ", v)
```

```
## [1] 6 3
## attr(,"match.length")
## [1] 1 1
## attr(,"useBytes")
## [1] TRUE
```

If you compare the result with the vector elements, than 6 and 3 gives the position of the blank within each of the two strings. Since we acutally want not the second word including a leading blank but only the second one, we have to increase the starting position by one.

```
regexpr(" ", v) + 1
```

```
## [1] 7 4
## attr(,"match.length")
## [1] 1 1
## attr(,"useBytes")
## [1] TRUE
```

Hence, the problem of the starting position is solved. As ending position, we use the result of the `nchar` function which returns the number of characters in a string.

```
nchar(v)
```

```
## [1] 11  8
```

In summary, this is one possible solution:

```
substring(v, regexpr(" ", v)+1, nchar(v))
```

```
## [1] "Earth" "Pluto"
```

For more informatino on the `regexpr` and similar functions, have a look at the help page (i.e. `?regexpr`).

For an overview of regular expressions which e.g. match any digit or alphabetic character see `?regex`.