Seminar I

1. Find all the empty lines from a file:

```
grep -E "^$" file
```

2. Find all the entries from the /etc/passwd file that have at least 22 vowels on the 5th field:

```
grep   -E   -i   "^([^:]*:){4}([^aeiou]*[aeiou][^aeiou]*){22,}(:[^:]*){2}$"
/etc/passwd
```

- /etc/passwd has a fixed separator (specifically the ":" character), so we have to write a regular expression that matches anything that may appear in the first 4 fields of the file and then write the regular expression that matches the given number of vowels
- `[^:]*:` -> this can be described as any amount of characters that are not ":" (the separator) followed by ":"; effectively this matches any string from a field from file /etc/passwd
- we repeat the previous expression 4 times, and thus we cover the first 4 fields of /etc/passwd
- `[^aeiou]*[aeiou][^aeiou]*` - this matches a vowel "wedged" in between other characters; this is needed since we are looking for non-adjacent vowels
- we repeat the previous structure at least 22 times -> this will match any string with at least 22 vowels, regardless if they are adjacent or not
- `:[^:]*` -> similar to the beginning, we create a pattern that matches a separator and the content of a field and we repeat it 2 times since that's how many fields are left in a line from /etc/passwd

3. Write a sed command that removes all fields except the 5th one from the /etc/passwd file.

```
sed -E "s/^(([^:]*:){4})([^:]*)((:[^:]*)*)/\3/" /etc/passwd
```

- the regular expressions are nearly identical as in problem nr. 2
- adding parentheses creates groups which are numbered automatically (in order of the opened parentheses) - i.e. `(([^:]*:){4})` is group nr 1 while `([^:]*:)` is group nr 2, etc.
- like the previous problem, `^(([^:]*:){4})` matches the first 4 fields starting from the beginning of the line
- `([^:]*)` this matches the contents of a field, minus the separators; added next to the previous expression, it will match exactly the contents of the 5th field (counting the number of opened parentheses in the regex so far, this would be group nr 3)
- `((:[^:]*)*)` this will represent groups 4 and 5 respectively (though in this particular case, it's meaningless to group them since we want to remove them) and will match all the fields after the 5th one
- `\3` references the 3rd group that is matched by the written regular expression, which is the 5th field of /etc/passwd

4. Write a sed command to swap consecutive lowercase letters

```
sed -E -i "s/([a-z])([a-z])/\2\1/g" file
```

5. Write a sed command that replaces every occurrence of “ab” with “ba” and every occurrence of “ba” with “ab”

```
sed -E 's/((a)(b))|((b)(a))/\3\2\6\5/g'
```

- `((a)(b))|((b)(a))` - the “|” character is the *or* operator, so this expression will match either *ab* or *ba*
- in this regular expression, there are a total of 6 groups defined, but only half of them will have a non empty value at any given time
- if the regular expression matches an “ab” sequence, then \1 will be “ab”, \2 will be “a”, and \3 will be “b”; groups \4, \5, and \6 will be empty (but they will be defined, since trying to use a group number that is not defined will result in an error)
- similarly, if the regular expression matches a “ba”, then \1, \2, and \3 will be empty and \5 will be “b” and \6 will be “a”

6. From a comma-separated values file with 2 fields, extract the first 2 consecutive letters from the first field, combine them with the contents of the second field, and create email addresses ending with “.com”. If there aren’t at least 2 consecutive letters in the first field of a line, leave the line unchanged.

```
sed -E "s/^[^a-z,]*([a-z]{2})[^,]*,([^,]*)$/\1@\2.com/" file
```

- `[^a-z,]*` - this will match any non-letter and non-comma character, repeated any number of times
- `([a-z]{2})` - two consecutive letters
- `[^,]*` - any character except the separator
- the three previous expressions, put together, will match a string containing at least 2 consecutive letters and will group the 2 consecutive letters in group nr 1
- `,([^,]*)` - this will match the separator and a string containing anything except the separator, so, in short, the second field and groups it in group nr 2
- `\1@\2.com` - creates a string containing the first 2 consecutive letters, followed by “@” and then adds the second group and forces a “.com” at the end