REGEX special characters

.	Matches any single character
\	Escape, changes the meaning of the character following it, between normal and special
[abc]	Matches any single character that appears in the list (in this case a or b or c)
[a-z]	Matches any single character that belongs to the range (in this case any lower-case letter)
[^0-9]	Matches any single character that does not belong to the range (in this case anything that is not a digit)
^	Beginning of line
$	End of line
\<	Beginning of word
\>	End of word
( )	Group several characters into an expression
*	Previous expression zero or more times
+	Previous expression one or more times
?	Previous expression zero or one times
{m,n}	Previous expression at least m and at most n times
|	Logical OR between parts of the regular expression

1. Print only the first 4 fields from each even-numbered line from a file, considering that the fields are separated by whitespaces. If a line has fewer than 4 fields, print all the fields from that line.

```
awk 'NR % 2 == 0 {print $1" "$2" "$3" "$4}' test_input_1
```

2. Print all the lines that contain only non-alphanumeric characters from a file. (any character that isn't a letter or a digit).

```
grep -E "^[^a-zA-Z0-9]*$" test_input_1
```
or
```
grep -E -v "[a-zA-Z0-9]" test_input_1
```

3. Duplicate each occurrence of an integer number from a file. We will consider that an integer number is a sequence of neighboring base 10 digits.
    - Ex: line "This 1234 is a number" will become "This 12341234 is a number"
    - Ex: line "56.34" will become "5656.3434"

```
sed -E "s/([0-9]+)/\1\1/g" test_input_1
```

4. Delete all characters after the last whitespace from each line from a file.
    - Ex: line *"A regular, boring line"* will become "*A regular, boring "*
    - Ex: line *"A less regular line"* will become *"A less regular "*

```
sed -E "s/(\s)[^[:space:]]*$/\1/" test_input_2
```
Comment: if a "match any character except" is needed for some characters represented by a backslash expression (like \s in this case) [^\s] will not yield the desired result. [^\s] will mean "any character except \ and s". So we need to use a character class -> [:space:].
See **man grep**, section "REGULAR EXPRESSION", subsection "Character Classes and Bracket Expressions".

5. Print the line number and the field from the middle of the line from each line that contains an odd number of fields from a file. Consider that the fields are separated by whitespaces. Note: division in awk is by default float division. If you need the integer part of a division use the int function. Ex: int(5/2) = 2.

```
awk 'NF % 2 == 1{i=int((NF+1)/2); print NR" "$i}' test_input_1
```

6. Swap field number 2 with field number 3 from a file where the fields are separated by the "**:**" character (Ex. /etc/passwd if available, but any file where fields are separated by : should do)

```
sed -E "s/^([^:]*):([^:]*):([^:]*)/\1:\3:\2/" /etc/passwd
```

7. Print all lines that contain at most 5 vowels, not necessarily consecutive, situated between 2 **^** signs from a file.
    - Ex: line *"aei^, still works^"* satisfies the condition
    - Ex: line *"abc^, way too many vowels here ^"* has too many vowels between the two **^**

- Ex: line *“^here there are too many vowels^but not here^”* satisfies the condition because there are 4 vowels between the second and third occurrences of the **^** character

```
grep -E "\^([^aeiouAEIOU]*[aeiouAEIOU][^aeiouAEIOU]*){0,5}\^" test_input_4
```

8. Remove the first word containing only lowercase letters from each line of a file

```
sed -E "s/\<[a-z]+\>//" test_input_1
sed -E "s/\b[a-z]+\b//" test_input_1
sed -E "s/[[:<:]][a-z]+[[:>:]]\b//" test_input_1 - !! this maybe works on macOS, it will not work on the exam server !!
```

9. Print the number of processes run by each user in the system (in the format *nr_processes user*). You can obtain a list of all processes in the system and the user that is running each process using **ps -ef**. Check the manual for **sort** and **uniq**.

```
ps -ef | grep -E -v "^UID" | awk '{print $1}' | sort | uniq -c
```

10. For each file from the current directory, display only the name of the file and the permissions for the user. (not the permissions for the *group* or for *other*; you can use **ls -l** to get information about files and folders from the current directory)

```
ls -l | grep -E "^-" | awk '{print $1" "$NF}' | sed -E "s/^.(...).{7}/\1/"
```

11. Display the number of lines that end in a vowel and the number of lines that end in a consonant from a file.

```
awk -f file.awk input.txt
```

Where the contents of `file.awk` are:

```
BEGIN {
     c=0;
     v=0;
}
$0 ~ /[aeiouAEIOU]$/ {
     v++;
}
$0 ~ /[a-zA-Z]$/ && $0 ~ /[^aeiouAEIOU]$/ {
     c++;
}
END {
     print "Number of lines ending with vowel: "v;
     print "Number of lines ending with consonant: "c;
}
```

12. Calculate the sum of all PIDs (process IDs) that use one of the editors: vim, joe, emacs, nano, pico. Use ps -ef or ps aux to get a list of all the existing processes from the system.

```
ps -ef | grep -E "^([^ ]+[ ]+){7}(vim|joe|nano|pico|emacs)" | awk 'BEGIN
{sum=0;} {sum+=$2;} END{print "Sum: "sum}'
```

13. Display all the users in the system (from the /etc/passwd file) whose username starts with a vowel.

```
grep -E -i "^[aeiou]" /etc/passwd
```

14. Display all the regular files from the current folder that have read permissions for the owner, group and other.

```
ls -l | grep -E "^-(r..){3}"
```

15. Using the last command display how many times each user has logged in the system.

```
last | awk '{print $1}' | sort | uniq -c | sort -rn
```

16. Using only grep, display the number of lines in a file.

```
grep -E -c ".*" input.txt
```

17. Determine the number of duplicate lines in a file.

```
sort input_file | uniq -c | awk 'BEGIN{c=0} $1 > 1 {c++} END {print "There
are " c " duplicated lines"}'
```

18. Display all the regular files smaller than 100 bytes from a given directory (non-recursively).

```
ls -l dir | awk '$1 ~ /^-/ {print $NF}'
```

19. Display all the usernames in a system but replace any digits in the username with the digit + 1 (except for 9, which will be replaced by 0).

```
cut -d: -f1 /etc/passwd | sed "y/0123456789/1234567890/"
```

20. Calculate the sum of PIDs of the processes running in the system for each username containing up to 4 characters.

```
f.awk:

NR > 1 && $1 ~ /[A-Za-z0-9]{,4}/ {
     if (!($1 in suma)) {
          suma[$1] = 0;
```

```
     }
     suma[$1] += $2;
}

END {
     for (i in suma) {
          print i" "suma[i];
     }
}

ps -ef | awk -f f.awk --posix
```

The --posix option is added to enable the usage of interval expression (ie. {4})

21. Using awk, display every other field for each line of a file. (ie. print only the 1st, 3rd, 5th, etc field)

```
awk '{for(i=2; i<=NF; i+=2){ $i="" } print $0}' input
```