

# Ćwiczenie 5

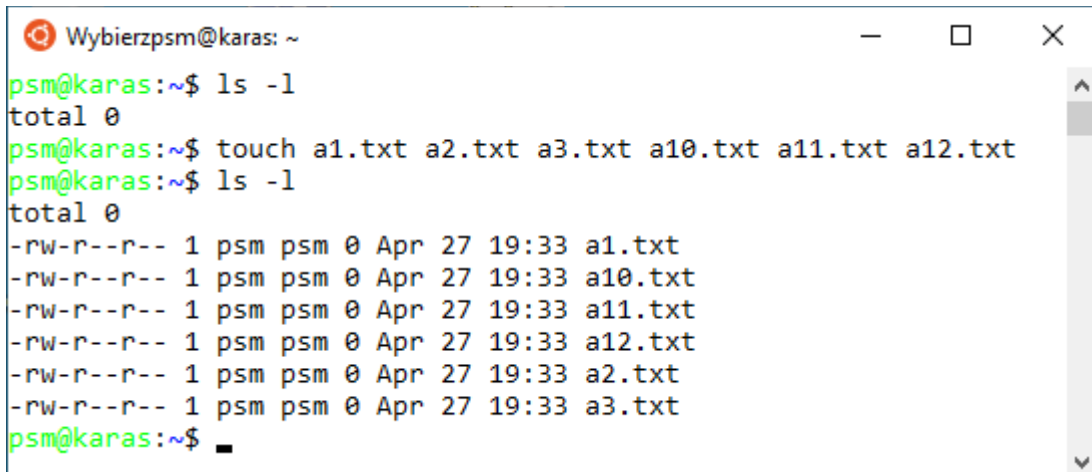
## Skrypty dla bash-a

### Cel ćwiczenia:

Przedmiotem ćwiczenia jest przedstawienie możliwości napisania skryptów w środowisku bash-a.

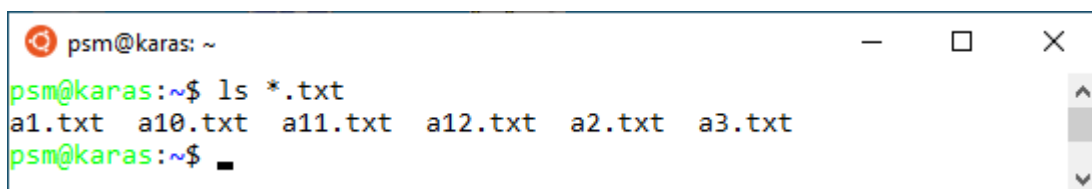
### Przebieg ćwiczenia - zagadnienia:

1. Przeanalizujemy następujące przykłady:
  - a. przygotujemy puste pliki do ćwiczeń:



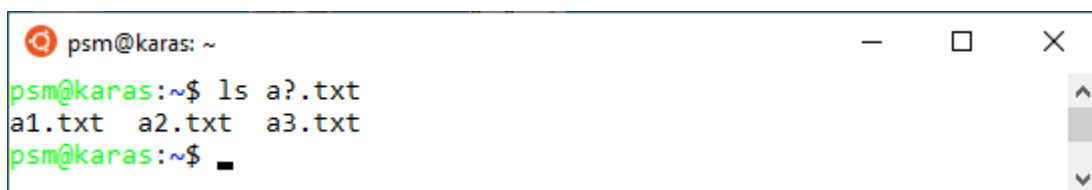
```
Wybierzpsm@karas: ~  
psm@karas:~$ ls -l  
total 0  
psm@karas:~$ touch a1.txt a2.txt a3.txt a10.txt a11.txt a12.txt  
psm@karas:~$ ls -l  
total 0  
-rw-r--r-- 1 psm psm 0 Apr 27 19:33 a1.txt  
-rw-r--r-- 1 psm psm 0 Apr 27 19:33 a10.txt  
-rw-r--r-- 1 psm psm 0 Apr 27 19:33 a11.txt  
-rw-r--r-- 1 psm psm 0 Apr 27 19:33 a12.txt  
-rw-r--r-- 1 psm psm 0 Apr 27 19:33 a2.txt  
-rw-r--r-- 1 psm psm 0 Apr 27 19:33 a3.txt  
psm@karas:~$
```

- b. gwiazdka oznacza cokolwiek:



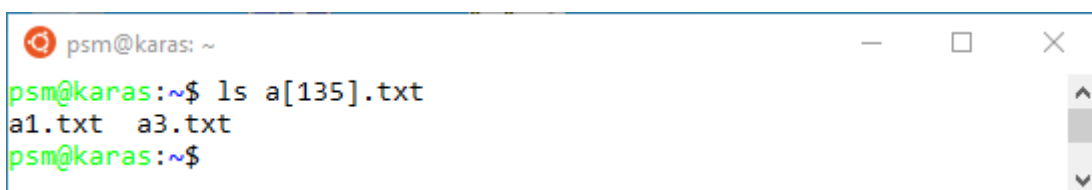
```
psm@karas: ~  
psm@karas:~$ ls *.txt  
a1.txt a10.txt a11.txt a12.txt a2.txt a3.txt  
psm@karas:~$
```

- c. znak zapytania to dokładnie jeden znak do dopasowania



```
psm@karas: ~  
psm@karas:~$ ls a?.txt  
a1.txt a2.txt a3.txt  
psm@karas:~$
```

- d. dowolny pojedynczy znak z listy zawartej pomiędzy nawiasami kwadratowymi – para nawiasów kwadratowych definiuje dokładnie jeden znak – 1 3 5 na drugim miejscu w nazwie:



```
psm@karas: ~  
psm@karas:~$ ls a[135].txt  
a1.txt a3.txt  
psm@karas:~$
```

- e. dowolny pojedynczy znak z listy zawartej pomiędzy nawiasami kwadratowymi – para nawiasów kwadratowych definiuje dokładnie jeden znak – zaprzeczenie – nie może być 1 3 5 na drugim miejscu w nazwie:

```
psm@karas: ~  
psm@karas:~$ ls a[!135].txt  
a2.txt  
psm@karas:~$
```

f. dowolny pojedynczy znak z podanego zakresu od 1 do 4 na drugim miejscu w nazwie:

```
psm@karas: ~  
psm@karas:~$ ls a[1-4].txt  
a1.txt a2.txt a3.txt  
psm@karas:~$
```

g. połączenie zakresu z listą możliwości – 1 lub zakres od 3 do 5 na drugim miejscu w nazwie:

```
psm@karas: ~  
psm@karas:~$ ls a[13-5].txt  
a1.txt a3.txt  
psm@karas:~$
```

h. przygotujmy kolejną porcję plików z pomocą polecenia touch

```
psm@karas: ~  
psm@karas:~$ touch g1.txt g2.txt g3.txt g10.txt g11.txt g12.txt  
psm@karas:~$ touch t1.txt t2.txt t3.txt t10.txt t11.txt t12.txt  
psm@karas:~$ ls -l  
total 0  
-rw-r--r-- 1 psm psm 0 Apr 27 19:33 a1.txt  
-rw-r--r-- 1 psm psm 0 Apr 27 19:33 a10.txt  
-rw-r--r-- 1 psm psm 0 Apr 27 19:33 a11.txt  
-rw-r--r-- 1 psm psm 0 Apr 27 19:33 a12.txt  
-rw-r--r-- 1 psm psm 0 Apr 27 19:33 a2.txt  
-rw-r--r-- 1 psm psm 0 Apr 27 19:33 a3.txt  
-rw-r--r-- 1 psm psm 0 Apr 27 20:07 g1.txt  
-rw-r--r-- 1 psm psm 0 Apr 27 20:07 g10.txt  
-rw-r--r-- 1 psm psm 0 Apr 27 20:07 g11.txt  
-rw-r--r-- 1 psm psm 0 Apr 27 20:07 g12.txt  
-rw-r--r-- 1 psm psm 0 Apr 27 20:07 g2.txt  
-rw-r--r-- 1 psm psm 0 Apr 27 20:07 g3.txt  
-rw-r--r-- 1 psm psm 0 Apr 27 20:08 t1.txt  
-rw-r--r-- 1 psm psm 0 Apr 27 20:08 t10.txt  
-rw-r--r-- 1 psm psm 0 Apr 27 20:08 t11.txt  
-rw-r--r-- 1 psm psm 0 Apr 27 20:08 t12.txt  
-rw-r--r-- 1 psm psm 0 Apr 27 20:08 t2.txt  
-rw-r--r-- 1 psm psm 0 Apr 27 20:08 t3.txt  
psm@karas:~$
```

i. połączenie zakresu z listą możliwości – zakres od a do h na pierwszym miejscu i cyfra 1 na drugim miejscu w nazwie:

```
psm@karas: ~  
psm@karas:~$ ls [a-h]1.txt  
a1.txt g1.txt  
psm@karas:~$
```

- j. połączenie zakresu z dwoma listami możliwości – zakres od e do z na pierwszym miejscu oraz zakres od 2 do 3 na drugim miejscu w nazwie:

```
psm@karas: ~  
psm@karas:~$ ls [e-z][2-3].txt  
g2.txt g3.txt t2.txt t3.txt  
psm@karas:~$
```

2. Korzystanie z mechanizmu rozwijania nawiasów w bash { }  
a. wynik – trzy napisy nieposortowane

```
psm@karas: ~  
psm@karas:~$ echo napis{3,2,1}  
napis3 napis2 napis1  
psm@karas:~$
```

- b. utworzenie czterech pustych plików:

```
psm@karas: ~  
psm@karas:~$ ls -l  
total 0  
psm@karas:~$ touch przyklad_{A,B,C,D}  
psm@karas:~$ ls -l  
total 0  
-rw-r--r-- 1 psm psm 0 Apr 28 08:36 przyklad_A  
-rw-r--r-- 1 psm psm 0 Apr 28 08:36 przyklad_B  
-rw-r--r-- 1 psm psm 0 Apr 28 08:36 przyklad_C  
-rw-r--r-- 1 psm psm 0 Apr 28 08:36 przyklad_D  
psm@karas:~$
```

- c. utworzenie czterech podkatalogów:

```
psm@karas: ~  
psm@karas:~$ mkdir Katalog{pierwszy,drugi,trzeci,czwarty}  
psm@karas:~$ ls -l  
total 0  
drwxr-xr-x 1 psm psm 512 Apr 28 08:39 Katalogczwarty  
drwxr-xr-x 1 psm psm 512 Apr 28 08:39 Katalogdrugi  
drwxr-xr-x 1 psm psm 512 Apr 28 08:39 Katalogpierwszy  
drwxr-xr-x 1 psm psm 512 Apr 28 08:39 Katalogtrzeci  
-rw-r--r-- 1 psm psm 0 Apr 28 08:36 przyklad_A  
-rw-r--r-- 1 psm psm 0 Apr 28 08:36 przyklad_B  
-rw-r--r-- 1 psm psm 0 Apr 28 08:36 przyklad_C  
-rw-r--r-- 1 psm psm 0 Apr 28 08:36 przyklad_D  
psm@karas:~$
```

- d. pliki i katalogi zawierające w nazwie string wraz ze znakami uogólniającymi:

```
psm@karas: ~  
psm@karas:~$ ls *{kl,gp}*  
przyklad_A przyklad_B przyklad_C przyklad_D  
  
Katalogpierwszy:  
psm@karas:~$
```

3. Przykład skryptu wyświetlającego elementy zadanej tablicy:

- a. treść zadania:

```
psm@karas: ~  
psm@karas:~$ cat skrypt1  
# Napisz skrypt, który wypisze listę elementów zadanej tablicy  
# oraz wszystkie elementy zadanej tablicy  
#  
psm@karas:~$
```

- b. treść skryptu

```
psm@karas: ~  
psm@karas:~$ cat skrypt1  
# Napisz skrypt, który wypisze listę elementów zadanej tablicy  
# oraz wszystkie elementy zadanej tablicy  
#  
tab=(pn wt sr czw pt sob nie)  
echo Liczba elementów tablicy: ${#tab[@]}  
a=0  
until [ $a -eq ${#tab[@]} ] ; do  
    echo tab[$a] = ${tab[$a++]}  
done  
  
psm@karas:~$
```

- c. wynik działania:

```
psm@karas: ~  
psm@karas:~$ ./skrypt1  
Liczba elementów tablicy: 7  
tab[0] = pn  
tab[1] = wt  
tab[2] = sr  
tab[3] = czw  
tab[4] = pt  
tab[5] = sob  
tab[6] = nie  
psm@karas:~$
```

4. Napisać skrypt o treści jak poniżej:

- a. treść zadania:

```
psm@karas: ~  
psm@karas:~$ cat skrypt2  
# Napisz skrypt, który za każdym razem dokonuje czynności losowania  
# zadanej czynności z listy czynności znajdujących się w predefiniowanej  
# tablicy  
# zmienna $RANDOM generuje całkowitą liczbę losową z zakresu 0 - 32767  
# Należy pamiętać, że rozkład tego generatora całkowitych liczb losowych  
# nie jest równomierny - przykład:  
# dla ${#czynnosc} = 30000 wyrażenie $RANDOM % ${#czynnosc[@]}  
# wylosuje elementy od 0 do 2767 dwa razy częściej niż pozostałe  
# elementy.
```

b. treść skryptu:

```
Wybierzpsm@karas: ~  
psm@karas:~$ cat skrypt2  
# Napisz skrypt, który za każdym razem dokonuje czynności losowania  
# zadanej czynności z listy czynności znajdujących się w predefiniowanej  
# tablicy  
# zmienna $RANDOM generuje całkowitą liczbę losową z zakresu 0 - 32767  
# Należy pamiętać, że rozkład tego generatora całkowitych liczb losowych  
# nie jest równomierny - przykład:  
# dla ${#czynnosc} = 30000 wyrażenie $RANDOM % ${#czynnosc[@]}  
# wylosuje elementy od 0 do 2767 dwa razy częściej niż pozostałe  
# elementy.  
#  
czynnosc=(kino spacer klub wyklad)  
k=$(( $RANDOM % ${#czynnosc[@]} ))  
echo "Teraz: ${czynnosc[$k]}"  
  
psm@karas:~$
```

c. wynik działania:

```
psm@karas: ~  
psm@karas:~$ ./skrypt2  
Teraz: klub  
psm@karas:~$ ./skrypt2  
Teraz: kino  
psm@karas:~$ ./skrypt2  
Teraz: spacer  
psm@karas:~$ ./skrypt2  
Teraz: kino  
psm@karas:~$ ./skrypt2  
Teraz: klub  
psm@karas:~$ ./skrypt2  
Teraz: kino  
psm@karas:~$ ./skrypt2  
Teraz: kino  
psm@karas:~$ ./skrypt2  
Teraz: spacer  
psm@karas:~$
```

5. Napisać skrypt o treści jak poniżej:

a. treść zadania:

```
psm@karas: ~  
psm@karas:~$ cat skrypt3  
# nazwa ()  
# {  
#     instrukcje  
# }  
# Przykład: napisz podprogram obliczania silni  
#
```

b. treść skryptu:

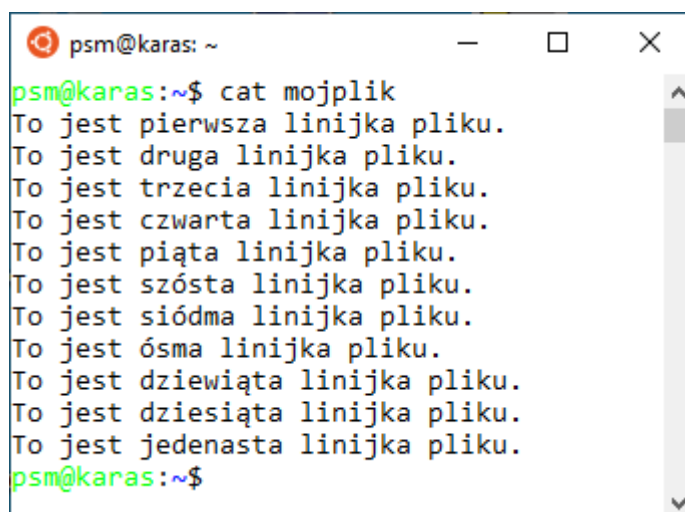
```
Wybierzpsm@karas: ~  
psm@karas:~$ cat skrypt3  
# nazwa ()  
# {  
#     instrukcje  
# }  
# Przykład: napisz podprogram obliczania silni  
#  
silnia ()  
{  
if [ $1 == 0 ] ; then  
    WYNIK=1  
else  
    silnia `expr $1 - 1`  
    WYNIK=`expr $WYNIK \* $1`  
    echo Argument: $1 WYNIK: $WYNIK  
fi  
}  
# Wywołanie  
silnia $1  
echo Wynik końcowy: $WYNIK  
psm@karas:~$
```

c. wynik działania:

```
psm@karas: ~  
psm@karas:~$ ./skrypt3 0  
Wynik końcowy: 1  
psm@karas:~$ ./skrypt3 1  
Argument: 1 WYNIK: 1  
Wynik końcowy: 1  
psm@karas:~$ ./skrypt3 2  
Argument: 1 WYNIK: 1  
Argument: 2 WYNIK: 2  
Wynik końcowy: 2  
psm@karas:~$ ./skrypt3 3  
Argument: 1 WYNIK: 1  
Argument: 2 WYNIK: 2  
Argument: 3 WYNIK: 6  
Wynik końcowy: 6  
psm@karas:~$ ./skrypt3 4  
Argument: 1 WYNIK: 1  
Argument: 2 WYNIK: 2  
Argument: 3 WYNIK: 6  
Argument: 4 WYNIK: 24  
Wynik końcowy: 24  
psm@karas:~$
```

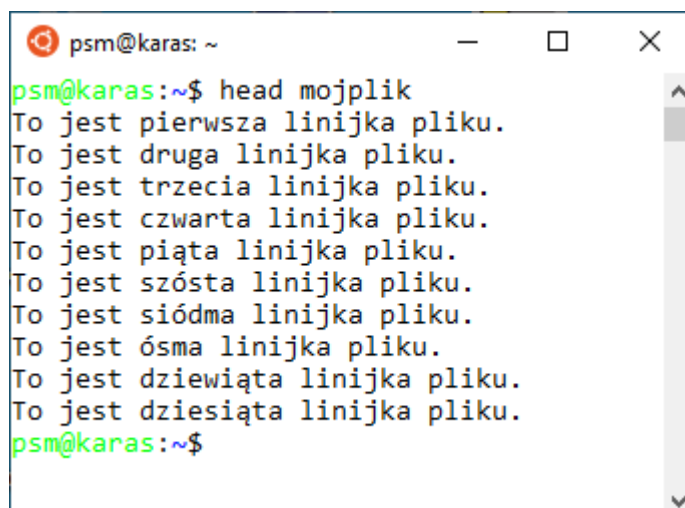
6. Przeanalizujmy poniższe polecenia:

a. przygotujmy plik o zawartości jak poniżej:



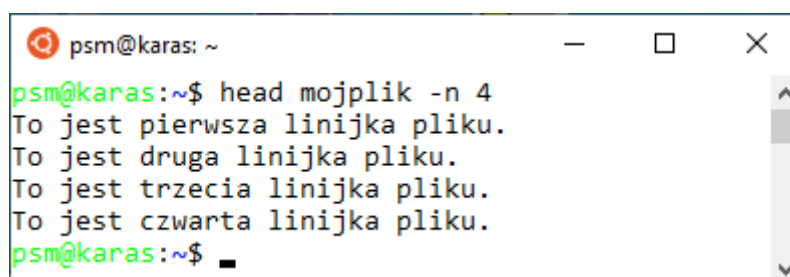
```
psm@karas: ~  
psm@karas:~$ cat mojplik  
To jest pierwsza linijka pliku.  
To jest druga linijka pliku.  
To jest trzecia linijka pliku.  
To jest czwarta linijka pliku.  
To jest piąta linijka pliku.  
To jest szósta linijka pliku.  
To jest siódma linijka pliku.  
To jest ósma linijka pliku.  
To jest dziewiąta linijka pliku.  
To jest dziesiąta linijka pliku.  
To jest jedenasta linijka pliku.  
psm@karas:~$
```

b. polecenie head wypisuje 10 pierwszych wierszy:



```
psm@karas: ~  
psm@karas:~$ head mojplik  
To jest pierwsza linijka pliku.  
To jest druga linijka pliku.  
To jest trzecia linijka pliku.  
To jest czwarta linijka pliku.  
To jest piąta linijka pliku.  
To jest szósta linijka pliku.  
To jest siódma linijka pliku.  
To jest ósma linijka pliku.  
To jest dziewiąta linijka pliku.  
To jest dziesiąta linijka pliku.  
psm@karas:~$
```

c. polecenie head z opcją -n iadaną wartością wypisuje ilość linii (od początku) zadaną tą wartością:



```
psm@karas: ~  
psm@karas:~$ head mojplik -n 4  
To jest pierwsza linijka pliku.  
To jest druga linijka pliku.  
To jest trzecia linijka pliku.  
To jest czwarta linijka pliku.  
psm@karas:~$
```

d. wykonajmy kopię pliku mojplik

```
psm@karas: ~  
psm@karas:~$ cp mojplik mojplik2  
psm@karas:~$ ls -l  
total 0  
-rw-r--r-- 1 psm psm 345 Apr 29 09:18 mojplik  
-rw-r--r-- 1 psm psm 345 Apr 29 09:24 mojplik2  
psm@karas:~$
```

e. wydajmy teraz polecenie `head *`

```
psm@karas: ~  
psm@karas:~$ head *  
==> mojplik <==  
To jest pierwsza linijka pliku.  
To jest druga linijka pliku.  
To jest trzecia linijka pliku.  
To jest czwarta linijka pliku.  
To jest piąta linijka pliku.  
To jest szósta linijka pliku.  
To jest siódma linijka pliku.  
To jest ósma linijka pliku.  
To jest dziewiąta linijka pliku.  
To jest dziesiąta linijka pliku.  
  
==> mojplik2 <==  
To jest pierwsza linijka pliku.  
To jest druga linijka pliku.  
To jest trzecia linijka pliku.  
To jest czwarta linijka pliku.  
To jest piąta linijka pliku.  
To jest szósta linijka pliku.  
To jest siódma linijka pliku.  
To jest ósma linijka pliku.  
To jest dziewiąta linijka pliku.  
To jest dziesiąta linijka pliku.  
psm@karas:~$
```

f. a teraz wydajmy polecenie `head * -q`



```
psm@karas: ~  
psm@karas:~$ head * -q  
To jest pierwsza linijka pliku.  
To jest druga linijka pliku.  
To jest trzecia linijka pliku.  
To jest czwarta linijka pliku.  
To jest piąta linijka pliku.  
To jest szósta linijka pliku.  
To jest siódma linijka pliku.  
To jest ósma linijka pliku.  
To jest dziewiąta linijka pliku.  
To jest dziesiąta linijka pliku.  
To jest pierwsza linijka pliku.  
To jest druga linijka pliku.  
To jest trzecia linijka pliku.  
To jest czwarta linijka pliku.  
To jest piąta linijka pliku.  
To jest szósta linijka pliku.  
To jest siódma linijka pliku.  
To jest ósma linijka pliku.  
To jest dziewiąta linijka pliku.  
To jest dziesiąta linijka pliku.  
psm@karas:~$
```

Jaka jest różnica pomiędzy efektami tych dwóch poleceń?

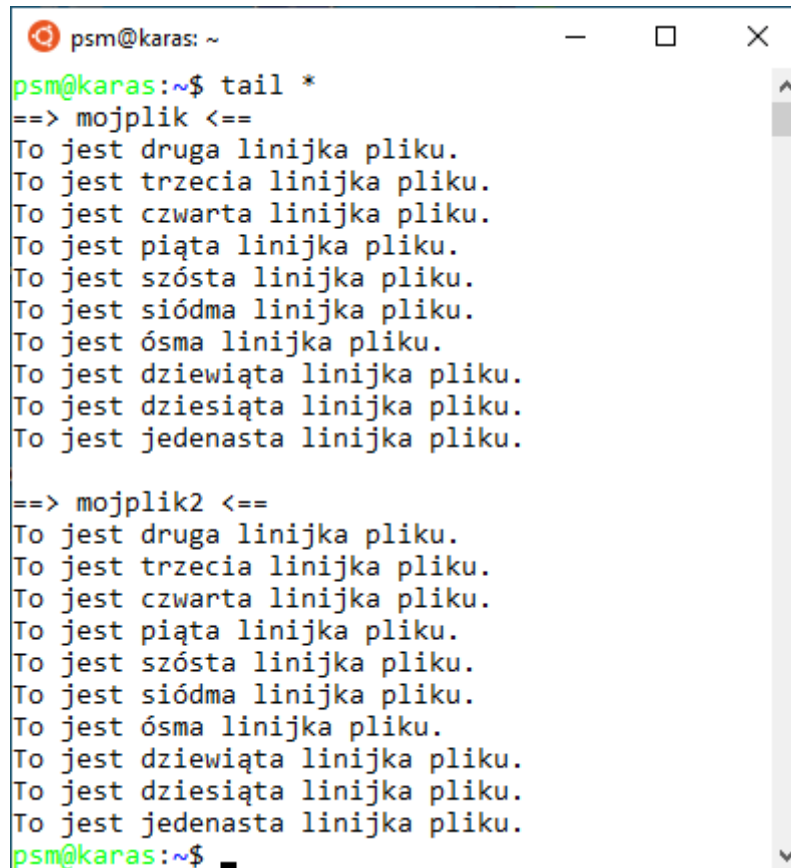
- g. wydamy teraz polecenie `tail` wypisuje ono 10 ostatnich linii:

```
psm@karas: ~  
psm@karas:~$ tail mojplik  
To jest druga linijka pliku.  
To jest trzecia linijka pliku.  
To jest czwarta linijka pliku.  
To jest piąta linijka pliku.  
To jest szósta linijka pliku.  
To jest siódma linijka pliku.  
To jest ósma linijka pliku.  
To jest dziewiąta linijka pliku.  
To jest dziesiąta linijka pliku.  
To jest jedenasta linijka pliku.  
psm@karas:~$
```

- h. polecenie `tail` z opcją `-n` i zadaną wartością wypisuje ilość linii (od końca) zadaną tą wartością:

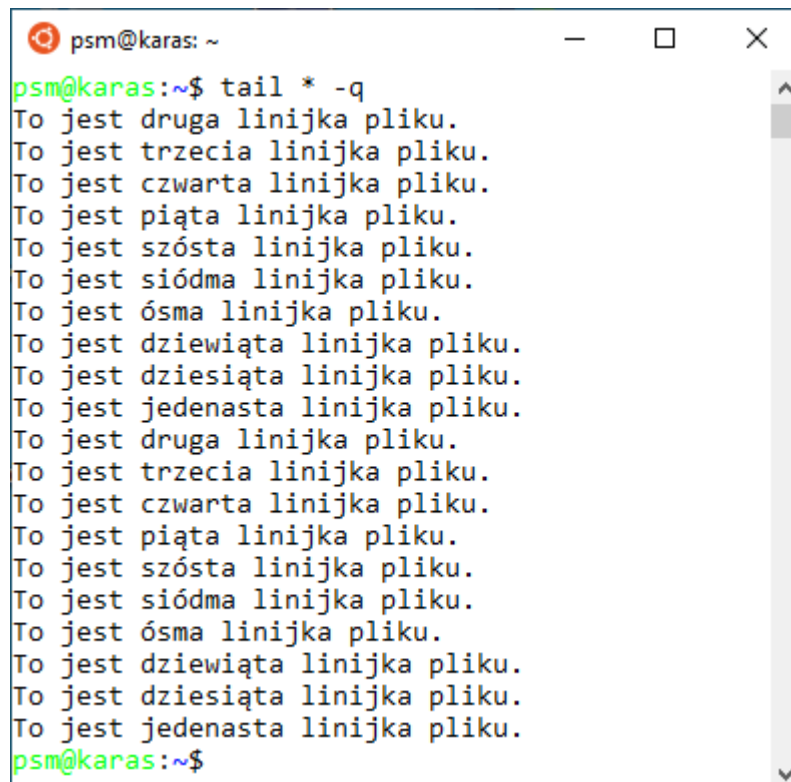
```
psm@karas: ~  
psm@karas:~$ tail mojplik -n 6  
To jest szósta linijka pliku.  
To jest siódma linijka pliku.  
To jest ósma linijka pliku.  
To jest dziewiąta linijka pliku.  
To jest dziesiąta linijka pliku.  
To jest jedenasta linijka pliku.  
psm@karas:~$
```

- i. teraz wydamy polecenie `tail *`



```
psm@karas: ~  
psm@karas:~$ tail *  
==> mojplik <==  
To jest druga linijka pliku.  
To jest trzecia linijka pliku.  
To jest czwarta linijka pliku.  
To jest piąta linijka pliku.  
To jest szósta linijka pliku.  
To jest siódma linijka pliku.  
To jest ósma linijka pliku.  
To jest dziewiąta linijka pliku.  
To jest dziesiąta linijka pliku.  
To jest jedenasta linijka pliku.  
  
==> mojplik2 <==  
To jest druga linijka pliku.  
To jest trzecia linijka pliku.  
To jest czwarta linijka pliku.  
To jest piąta linijka pliku.  
To jest szósta linijka pliku.  
To jest siódma linijka pliku.  
To jest ósma linijka pliku.  
To jest dziewiąta linijka pliku.  
To jest dziesiąta linijka pliku.  
To jest jedenasta linijka pliku.  
psm@karas:~$
```

- j. a teraz wydajmy polecenie `tail * -q`



```
psm@karas: ~  
psm@karas:~$ tail * -q  
To jest druga linijka pliku.  
To jest trzecia linijka pliku.  
To jest czwarta linijka pliku.  
To jest piąta linijka pliku.  
To jest szósta linijka pliku.  
To jest siódma linijka pliku.  
To jest ósma linijka pliku.  
To jest dziewiąta linijka pliku.  
To jest dziesiąta linijka pliku.  
To jest jedenasta linijka pliku.  
To jest druga linijka pliku.  
To jest trzecia linijka pliku.  
To jest czwarta linijka pliku.  
To jest piąta linijka pliku.  
To jest szósta linijka pliku.  
To jest siódma linijka pliku.  
To jest ósma linijka pliku.  
To jest dziewiąta linijka pliku.  
To jest dziesiąta linijka pliku.  
To jest jedenasta linijka pliku.  
psm@karas:~$
```

- k. zobaczmy jak działa polecenie `cut -c 4-9 mojplik` polecenie to wycina z każdej linijki pliku od 4 do 9 znaku

```
psm@karas: ~  
psm@karas:~$ cat mojplik  
To jest pierwsza linijka pliku.  
To jest druga linijka pliku.  
To jest trzecia linijka pliku.  
To jest czwarta linijka pliku.  
To jest piąta linijka pliku.  
To jest szósta linijka pliku.  
To jest siódma linijka pliku.  
To jest ósma linijka pliku.  
To jest dziewiąta linijka pliku.  
To jest dziesiąta linijka pliku.  
To jest jedenasta linijka pliku.  
psm@karas:~$ cut -c 4-9 mojplik  
jest p  
jest d  
jest t  
jest c  
jest p  
jest s  
jest s  
jest  
jest d  
jest d  
jest j  
psm@karas:~$
```

**UWAGA!** Tutaj widać dlaczego nie oplaca się używać polskich znaków diakrytycznych – zamiast ó mamy w jednej z linijek brak znaku (niekoniecznie musi to być spacja!).

1. zobaczymy jak działa polecenie `cut -d " " -f 3- mojplik` polecenie to wycina z każdej linijki pliku 3 pole do końca – separatorem pól jest spacja

```
psm@karas: ~  
psm@karas:~$ cut -d " " -f 3- mojplik  
pierwsza linijka pliku.  
druga linijka pliku.  
trzecia linijka pliku.  
czwarta linijka pliku.  
piąta linijka pliku.  
szósta linijka pliku.  
siódma linijka pliku.  
ósma linijka pliku.  
dziewiąta linijka pliku.  
dziesiąta linijka pliku.  
jedenasta linijka pliku.  
psm@karas:~$
```

- m. a teraz poleceniem `cut -d " " -f 2-3 mojplik` polecenie to wycina z każdej linijki pliku 2 i 3 pole – separatorem pól jest spacja

```
psm@karas: ~  
psm@karas:~$ cut -d " " -f 2-3 mojplik  
jest pierwsza  
jest druga  
jest trzecia  
jest czwarta  
jest piąta  
jest szósta  
jest siódma  
jest ósma  
jest dziewiąta  
jest dziesiąta  
jest jedenasta  
psm@karas:~$
```

- n. polecenie `grep` znajduje w pliku wiersze zgodne z podanym wyrażeniem:

```
psm@karas: ~  
psm@karas:~$ grep druga mojplik  
To jest druga linijka pliku.  
psm@karas:~$
```

- o. w poleceniu `grep` możemy stosować wyrażenia regularne

```
psm@karas: ~  
psm@karas:~$ grep "To jest "[ptcj] mojplik  
To jest pierwsza linijka pliku.  
To jest trzecia linijka pliku.  
To jest czwarta linijka pliku.  
To jest piąta linijka pliku.  
To jest jedenasta linijka pliku.  
psm@karas:~$
```

7. Przykład skryptu wykorzystującego wyrażenia regularne:

- a. treść zadania:

```
psm@karas: ~  
psm@karas:~$ cat skrypt4  
# Napisz skrypt, który zamienia nazwę miesiąca  
# w języku angielskim na nazwę miesiąca w języku polskim.  
# Nazwa miesiąca po angielsku jest podana jako argument  
# wywołania tego skryptu i może być napisana małymi lub dużymi literami -  
# dowolną ich kombinacją, np. JaNuARY  
#
```

- b. treść skryptu:

```
Wybierzpsm@karas: ~
psm@karas:~$ cat skrypt4
# Napisz skrypt, który zamienia nazwę miesiąca
# w języku angielskim na nazwę miesiąca w języku polskim.
# Nazwa miesiąca po angielsku jest podana jako argument
# wywołania tego skryptu i może być napisana małymi lub dużymi literami -
# dowolną ich kombinacją, np. JaNuARY
#
case $1 in
[Jj][Aa][Nn][Uu][Aa][Rr][Yy]) MIESIAC="styczeń" ;;
[Ff][Ee][Bb][Rr][Uu][Aa][Rr][Yy]) MIESIAC="luty" ;;
[Mm][Aa][Rr][Cc][Hh]) MIESIAC="marzec" ;;
[Aa][Pp][Rr][Ii][Ll]) MIESIAC="kwiecień" ;;
*) MIESIAC="nie rozpoznany" ;;
esac
echo WYNIK: $MIESIAC
psm@karas:~$
```

c. przykład działania:

```
psm@karas: ~
psm@karas:~$ ./skrypt4 Jan
WYNIK= nie rozpoznany
psm@karas:~$ ./skrypt4 JaNuARy
WYNIK= styczeń
psm@karas:~$ ./skrypt4 mArCH
WYNIK= marzec
psm@karas:~$ ./skrypt4 APRIL
WYNIK= kwiecień
psm@karas:~$
```

8. Napisać skrypt o treści jak poniżej:

a. treść zadania:

```
psm@karas: ~
psm@karas:~$ cat skrypt5
# Napisz skrypt, który pobierze cztery argumenty: PLIKWE1, PLIKWE2,
# PLIKWY1 oraz PLIKWY2 i wykorzystując wczytane pliki wejściowe
# o nazwach zadanych tymi argumentami PLIKWE1 oraz PLIKWE2 wykona
# następujące operacje:
# 1. Do pliku o nazwie zadanej argumentem PLIKWY1 naprzemiennie
#    przepisze kolejne linie z plików o nazwach zadanych argumentami
#    PLIKWE1 oraz PLIKWE2 w ten sposób, że:
#    a. jako pierwsza będzie pierwsza linia z pierwszego pliku,
#    b. jako druga linia będzie pierwsza linia z drugiego pliku,
#    c. jako trzecia linia będzie druga linia z pierwszego pliku,
#    d. jako czwarta linia będzie druga linia z drugiego pliku,
#    e. itp.
# 2. Jeżeli w dowolnym z plików wejściowych linie się skończą, to
#    pozostałe linie z dłuższego pliku wejściowego zostaną zapisane
#    do pliku o nazwie zadanej argumentem PLIKWY2.
# Skrypt powinien zawierać mechanizmy sprawdzenia poprawności
# uruchomienia tego skryptu: liczbę argumentów i istnienia plików
# wejściowych.
#
```

b. treść skryptu:

```
Wybierzpsm@karas: ~
#
ULPLIKWE1 ()
{
N1=0
LINIA1=""
while read LINIA ; do
    let N1++
    if [ $N1 -eq $NRL1 ] ; then
        LINIA1=$LINIA
    fi
done < "$PLIKWE1"
}
#
ULPLIKWE2 ()
{
N2=0
LINIA2=""
while read LINIA ; do
    let N2++
    if [ $N2 -eq $NRL2 ] ; then
        LINIA2=$LINIA
    fi
done < "$PLIKWE2"
}
#
przepSYM ()
{
if [ $LLPLIKWE1 -lt $LLPLIKWE2 ] ; then
    LLINII=$LLPLIKWE1
else
    LLINII=$LLPLIKWE2
fi
echo "liczba linii $LLINII"
for (( i=1 ; $i <= $LLINII ; i++ )) ; do
    NRL1=$i
    ULPLIKWE1
        echo "Znalazłem(1): $LINIA1"
    echo $LINIA1 >> "$PLIKWY1"
    NRL2=$i
    ULPLIKWE2
        echo "Znalazłem(2): $LINIA2"
    echo $LINIA2 >> "$PLIKWY1"
done
}
#
```

```
Wybierzpsm@karas: ~
#
niesymprzep ()
{
przepsym
if [ $LLPLIKWE1 -gt $LLPLIKWE2 ] ; then
    for (( i=$LLPLIKWE2+1 ; $i <= $LLPLIKWE1 ; i++ )) ; do
        NRL1=$i
        ULPLIKWE1
        echo "Znalazłem (1a): $LINIA1"
        echo $LINIA1 >> "$PLIKWY2"
    done
else
    for (( i=$LLPLIKWE1+1 ; $i <= $LLPLIKWE2 ; i++ )) ; do
        NRL2=$i
        ULPLIKWE2
        echo "Znalazłem (2a): $LINIA2"
        echo $LINIA2 >> "$PLIKWY2"
    done
fi
}
#
przepisanie ()
# porównanie plików wejściowych i decyzja
# czy są równe czy też nie
{
LLPLIKWE1=0
LLPLIKWE2=0
while read LINIA ; do
    let LLPLIKWE1++
done < "$PLIKWE1"
while read LINIA ; do
    let LLPLIKWE2++
done < "$PLIKWE2"
echo Plik "$PLIKWE1" ma $LLPLIKWE1 linii
echo Plik "$PLIKWE2" ma $LLPLIKWE2 linii
if [ $LLPLIKWE1 -eq $LLPLIKWE2 ] ; then
    echo Tryb pracy: przepisywanie symetryczne
    przepsym
else
    echo Tryb pracy: przepisywanie niesymetryczne
    niesymprzep
fi
}
```

```
Wybierzpsm@karas: ~
}
# Wywołanie
if [ $# -ne 4 ] ; then
    echo skrypt powinien byc wywołany z czterema argumentami
else
    if [ ! -f "$1" -o ! -f "$2" ] ; then
        echo "Plik(i) wejściowe o podanej(ych) nazwie(ach) nie istnieje(a)."
    else
        PLIKWE1=$1
        PLIKWE2=$2
        PLIKWY1=$3
        PLIKWY2=$4
        przepisanie
    fi
fi
psm@karas:~$
```

c. pliki danych:

```
psm@karas: ~
psm@karas:~$ cat PLIKWE1
PLIK 1 Linijka 1
PLIK 1 Linijka 2
psm@karas:~$ cat PLIKWE2
PLIK 2 Linijka 1
PLIK 2 Linijka 2
PLIK 2 Linijka 3
PLIK 2 Linijka 4
psm@karas:~$ █
```

d. wywołanie skryptu z monitorowaniem jego działania:

```
psm@karas: ~
psm@karas:~$ ./skrypt5 PLIKWE1 PLIKWE2 a b
Plik PLIKWE1 ma 2 linii
Plik PLIKWE2 ma 4 linii
Tryb pracy: przepisywanie niesymetryczne
liczba linii 2
Znalazłem(1): PLIK 1 Linijka 1
Znalazłem(2): PLIK 2 Linijka 1
Znalazłem(1): PLIK 1 Linijka 2
Znalazłem(2): PLIK 2 Linijka 2
Znalazłem (2a): PLIK 2 Linijka 3
Znalazłem (2a): PLIK 2 Linijka 4
psm@karas:~$
```

e. pliki wynikowe:



```
psm@karas: ~  
psm@karas:~$ cat a  
PLIK 1 Linijka 1  
PLIK 2 Linijka 1  
PLIK 1 Linijka 2  
PLIK 2 Linijka 2  
PLIK 1 Linijka 1  
PLIK 2 Linijka 1  
PLIK 1 Linijka 2  
PLIK 2 Linijka 2  
psm@karas:~$ cat b  
PLIK 2 Linijka 3  
PLIK 2 Linijka 4  
PLIK 2 Linijka 3  
PLIK 2 Linijka 4  
psm@karas:~$
```