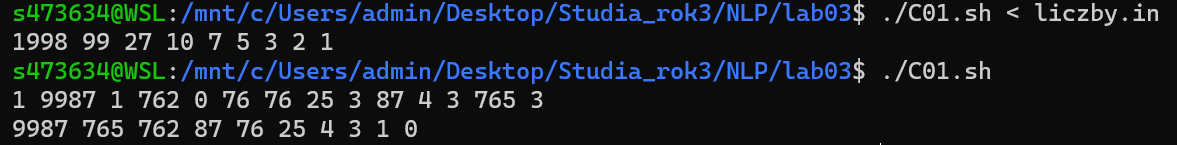
Miłosz Rolewski, s473634, Informatyka 3 rok studia dzienne

# C01



#!/bin/bash

# ./C01.sh < liczby.in

read numbers

numbers=$(echo $numbers | tr ' ' '\n')

sorted\_numbers=$(echo "$numbers" | sort -n -r -u)

echo $sorted\_numbers

Wejście: liczby w 1 linii oddzielone linią spacji  
Wyjście: liczby wejścia wypisane malejąco, bez powtórzeń

Komenda sort sortuje liczby wejścia. -n – po wartościach numerycznych, -r -malejąco/reverse,  
-u – unikalne wartości

# C02Obraz zawierający tekst, zrzut ekranu, Czcionka Opis wygenerowany automatycznie

#!/bin/bash

#linie mozna podac recznie lub z pliku auta komenda ./C02.sh < auta.in

max\_predkosc=0

najszybszy\_samochod=""

while IFS=, read -r marka model predkosc; do

    if (( $(echo "$predkosc > $max\_predkosc") )); then

        max\_predkosc="$predkosc"

        najszybszy\_samochod="$marka $model"

    fi

done

# echo "$najszybszy\_samochod"

echo $max\_predkosc

Wejście: uporządkowane trójki o elementach oddzielonych przecinkami  
Wyjście: prędkość najszybszego samochodu

Plik sczytuje kolejne linie z pliku/standardowego wejścia i przypisuje elementy trójki do poszczególnych zmiennych. Sprawdza czy obecna trójka ma większa prędkość niż dotychczas najwyższa, po czym jeżeli warunek jest spełniony nadpisuje dane. Po przejściu pętli wypisuje najwyższą znalezioną prędkość.

# C03

#!/bin/bash

n=123

declare -a lines

while IFS= read -r line; do

  lines+=("$line")

done

echo "${lines[n-1]}"

Linie można też wprowadzić ręcznie zamiast z pliku, ale wpisywanie 123 linii po enterze mija się z sensem :)  
Wejście: wiele linii… lub plik zawierający dużo linii  
Wyjście: linii nr 123, czyli o indeksie 122

W pętli while sczytuje linie do tablicy lines, a po jej przejściu wypisuje żądaną linię

# C04

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

n=10

m=20

declare -a lines

while IFS= read -r line; do

    lines+=("$line")

done

for ((i = n-1; i < m; i++)); do

    echo "$((i+1)): ${lines[i]}"

done

Wejście: wiele linii… lub plik zawierający dużo linii  
Wyjście: linie z zakresu 10-20

Sczytuje linie z pliku/standardowego wejścia aż nie natrafię na koniec pliku/wejścia, a następnie wyświetlam linie z podanego zakresu

# C05



#!/bin/bash

tmp\_file=$(mktemp)

while IFS= read -r line; do

    echo "$line" >> "$tmp\_file"

done

# echo "cz,sz count:"

grep -io "cz\|sz" "$tmp\_file" | wc -l

rm $tmp\_file

Wejście: plik/tekst zawierający polskie znaki  
Wyjście: ten sam tekst, lecz zamiast polskich znaków występuje symbol ‘X’

Sczytuje linie z pliku/standardowego wejścia aż nie napotkam na znak końca pliku, oraz wpisuje je do pliku tymczasowego. Komendą grep wyszukuje wzorca podanego wyrażeniem regularnym, czyli szukamy ‘sz’ i ‘cz’. Flaga -i ignoruje wielkość liter, a flaga -o powoduje, że szukamy wyłącznie w pod słowach, a nie liczymy wyłącznie całe wyrazy. Wc -l zlicza ilość które wyda grep

# C06

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

#!/bin/bash

tr -c 'A-Za-z0-9[:space:][:punct:]' 'X'

Wejście: plik/tekst zawierający polskie znaki  
Wyjście: ten sam tekst, w którym polskie znaki zamieniono na X

tr z flagą zamienia znaki z podanego zakresu na znak ‘X’, flaga -c sprawia, że odwracamy zasady klasyfikacji, czyli zostawiamy wyłącznie znaki z podanego wyrażenia regularnego, a wszystkie inne zmieniamy na znak ‘X’

# C07

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

#!/bin/bash

tmp\_file=$(mktemp)

i=1

while IFS= read -r line; do

    echo "$i $line" >> "$tmp\_file"

    ((i=i+1))

done

tac $tmp\_file > odwrotnie.out

rm $tmp\_file

Wejście: plik / tekst   
Wyjście: linie z wejścia wraz z indeksami wejściowymi w odwróconej kolejności

Sczytuje linie wejścia do pliku tymczasowego dopisując każdej z nich indeks. Po przejściu pętli odwracam plik tymczasowy komendą tac i przekierowuje wyjście do pliku odwrotnie.out. Na końcu usuwam plik tymczasowy.

# C08

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

#!/bin/bash

tr -d '[:punct:]' < interp.in > interp.out

wc -w < interp.out

Wejście: plik zawierający tekst z licznymi znakami interpunkcyjnymi  
Wyjście: ten sam tekst, pozbawiony znaków interpunkcyjnych, liczba słów w tekście bez znaków interpunkcyjnych

Polecenie tr -d usuwa wszystkie znaki pasujące do podanego wyrażenia regularnego, w tym przypadku wszystkie znaki interpunkcyjne. Wyjście przekierowuje do pliku interp.out po czym wywołuje na nim polecenie wc -w które zlicza ilość słów.

W razie problemów z archiwum plików, wszystkie skrypty i pliki wejściowe są dostępne na githubie:

https://github.com/milosz-amg/Natural\_language\_processing/tree/master/lab03