Pinchas Plevinski

322211558

Unix Lab 3 Bash and AWK (2023)

1. Write a Bash shell script which given a set of numbers on the command line, will print

them out sorted. **Do not use the already existing sort command.**

Here is a sample of how the program should work:

./mysort.sh 2 3 1 19 14 13 11 5 6 3 3

1 2 3 3 3 5 6 11 13 14 19

#!/bin/bash

nums=("$@") # storing the arguments in array

n=${#nums[@]} #size of the array

# bubble sort

for ((i = 0; i < n-1; i++)); do

for ((j = 0; j < n-i-1; j++)); do

if ((nums[j] > nums[j+1])); then

# Swap numbers

temp=${nums[j]}

nums[j]=${nums[j+1]}

nums[j+1]=$temp

fi

done

done

# printing the sorted array

for i in "${nums[\*]}"; do

echo "$i"

done

2. Write a shell script which imitates the wc command without using the wc command. The

program should handle the flags -l, -w, and -c. If an invalid flag is given, the program

should print “error.” If no flag is given, print all three details. Furthermore, the command

should accept several file names, in which case it will simply run on each file one after

the other printing the required details for each file.

#!/bin/bash

line\_flag=false

word\_flag=false

char\_flag=false

if [[ $1 =~ ^-.$ ]]; then

case $1 in

-l) line\_flag=true ;;

-w) word\_flag=true ;;

-c) char\_flag=true ;;

\*) echo "error" && exit 1 ;;

esac

fi

if [[ $1 != -\* ]]; then

line\_flag=true

word\_flag=true

char\_flag=true

fi

count\_lines\_words\_chars() {

line\_count=0

word\_count=0

char\_count=0

while read line; do

((line\_count++))

char\_count=$((char\_count + ${#line}))

words=($line)

word\_count=$((word\_count + ${#words[@]}))

done < $1

if $line\_flag; then

printf "%3d" $line\_count

fi

if $word\_flag; then

printf "%3d" $word\_count

fi

if $char\_flag; then

printf "%3d" $char\_count

fi

echo " $1"

}

total\_lines=0

total\_words=0

total\_chars=0

for file in "$@"; do

if [ -f "$file" ]; then

count\_lines\_words\_chars "$file"

counts=$(count\_lines\_words\_chars "$file")

read -r lines words chars filename <<< "$counts"

total\_lines=$((total\_lines + lines))

total\_words=$((total\_words + words))

total\_chars=$((total\_chars + chars))

fi

done

if $line\_flag; then

printf "%3d" $total\_lines

fi

if $word\_flag; then

printf "%3d" $total\_words

fi

if $char\_flag; then

printf "%3d" $total\_chars

fi

echo " total"

3. Create a file containing rows of comma delimited information where each row contains a

grade, a course number, and a student number in that order. Write a program (Bash or AWK or a combination) or a single command line operation which will print out the average grade for each course excluding all grades lower than 30. Do not assume the rows are in any order. They are shuffled randomly.

awk -F, '$1 >= 30 {sum[$2]+=$1; count[$2]++} END {for (course in sum) print "Course " course ": Average grade = " sum[course]/count[course]}' students\_file

4. Write an AWK script that reads a file and extracts any numbers in the file and checks if they are prime numbers.

The script should then print all prime numbers that it found.

Example file:

Math is179day foun193dary

18

nachum17

19

should print

179

193

17

19

To solve this you should either use the awk function **gsub** or combine your **awk with sed**.

#!/bin/bash

# Function to check if a number is prime

is\_prime() {

num="$1"

if [ "$num" -le 1 ]; then

return 1

fi

for ((i = 2; i \* i <= num; i++)); do

if [ "$((num % i))" -eq 0 ]; then

return 1

fi

done

return 0

}

awk '{

i = 1

while (i <= length($0)) {

char = substr($0, i, 1)

if (char ~ /[0-9]/) {

digit\_str = char

while ((i + 1) <= length($0) && substr($0, i + 1, 1) ~ /[0-9]/) {

digit\_str = digit\_str substr($0, i + 1, 1)

i++

}

# Add the digit\_str to the numbers array

numbers[digit\_str] = 1

}

i++

}

}

END {

# Print the contents of the numbers array

for (number in numbers) {

print number

}

}' $1 > .tmp

# Check each command-line argument

while read arg; do

if is\_prime "$arg"; then

echo $arg

fi

done < .tmp