Exam Prep Cheat Sheet

Topics to Study

- Read in a data file into a Bash script.
  - Use the read command to read in the contents of a file
  - Use redirection (`<`) to read in a file from the command line

```
# Using the read command
read filename
contents=$(cat $filename)
echo $contents

# Using redirection
contents=$(cat < $filename)
echo $contents
```

- Replace text in the data file.
  - Use the sed command to search and replace text in a file

```
sed 's/old_text/new_text/g' filename > newfile
```

- Read certain lines of the data file.
  - Use the head or tail command to read the first or last few lines of a file
  - Use a loop to iterate over the lines of a file and only read certain ones

```
# Using the head command
head -n 5 filename

# Using a loop
while read line; do
    if [some condition]; then
        echo $line
    fi
done < filename
```

- Look for certain columns in the data file.
  - Use the cut command to extract certain columns from a file
  - Use the awk command to search for specific columns based on certain criteria

```
# Using the cut command
cut -d ',' -f 2 filename

# Using the awk command
awk -F ',' '$1 == "value" {print $2}' filename
```

- Write/append to new files.
  - Use the echo command to write text to a file

    - Use the » operator to append text to an existing file

```
# Writing to a new file
echo "Hello, world!" > newfile

# Appending to an existing file
echo "Hello again!" >> existingfile
```

- Implement loops and conditionals.
    - Use the for loop to iterate over a list of items
    - Use the while loop to continue running a block of code while a condition is true
    - Use if/else statements to conditionally execute code

```
# Using a for loop
for i in {1..10}; do
    echo $i
done

# Using a for loop with seq
for i in $(seq 1 10); do
    echo $i
done

# Using a while loop
counter=1
while [ $counter -le 10 ]; do
    echo $counter
    ((counter++))
done

# Using if/else statements
if [ $a -gt $b ]; then
    echo "$a is greater than $b"
else
    echo "$b is greater than $a"
fi
```

- Write a function.
    - Use arguments to pass information to a function
    - Use the return statement to return a value from a function

```
greet() {
    echo "Hello, $1!"
}

greet "world"
```

Grading Rubric

- TeddyBallgame.csv is converted to TeddyBallgame.txt as specified: 10 points.

```
sed "1d" TeddyBallgame.csv | tr , " " > TeddyBallgame.txt
# or
head -n -1 TeddyBallgame.csv | tr ',' ' ' > TeddyBallgame.txt
# or
tail -n +2 TeddyBallgame.csv | tr ',' ' ' > TeddyBallgame.txt
```

- Script determines the number of lines in TeddyBallgame.txt

```
wc -l < TeddyBallgame.txt
```

- Script reads one line at a time from the .txt file

```
while read line; do
  echo "$line"
done < TeddyBallgame.txt
# or
for line in $(cat TeddyBallgame.txt)
do
  echo $line
done
```

- After a line is read, the remaining lines are output to temp.txt

```
counter=0
while read line; do
    echo "$line"
    sed "1,$counter d" TeddyBallgame.txt > temp.txt
    # or
    tail -n "+$counter" TeddyBallgame.txt > temp.txt

    (( count++ ))
    break # if necessary
done < TeddyBallgame.txt
```

- The contents of temp.txt are used to overwrite TeddyBallgame.txt after temp.txt is written

```
mv temp.txt TeddyBallgame.txt
```

- Script correctly checks for the specified statistical criteria

```
awk '$1 >= 30 {print $1,$2,$3}' TeddyBall.txt | uniq | sort -nrk2
```

- The line_writer function is implemented as specified (no arguments provided to the function)

```
line_writer() {
    local file="$1"
    local text="$2"

    # If the file exists
    if [[ -f "$file" ]]; then
        rm output.txt
        return 1
    fi

    # Or if the file doesn't exist
    if [[ ! -f $file ]]; then
        echo "File does not exist"
        return 1
    fi

    if [[ -z $line ]]; then
        echo "Line is empty or null"
        return 1
    fi

    echo "$line" >> "$file"
}
```

- Bonus: Script checks to see if output.txt already exists and deletes it if it does.

```
if [[ -f "output.txt" ]]; then
    rm output.txt
fi
```

Extra

```
# grep for string starts with ip or net
grep -E "^(ip|net).*|*.grep$"

# Output a list of all subdirectories of /etc that you cannot open.
# The list should only include the full path of the subdirectories without any extraneous v
find <folder> -type d -not -readable -printf "%f\n" 2>/dev/null | sed "s/^/\/<folder>\//g"
```