**Problem 1 (5 pts)**

1.Using man or google or chatGPT, look up what the inputs to echo are that we haven’t seen before. Comment each line of the code as to what it is intended to do.

# specifies that the script should be executed using the Bash shell.

#!/bin/bash

# echo is used to print text to the terminal, -n suppresses the trailing newline. prints '/', and -ne ensures no newline is added.

echo -ne '/';

# Introduce a delay after printing '/'. The 1 after sleep means a pause of 1 second.

sleep 1;

# Similar to the first echo line, this prints '-' and moves the cursor back one position (\b).

echo -ne '\b-';

# Another pause for 1 second.

sleep 1;

# Printing '\' and moving the cursor back.

echo -ne '\b\\'

# Another pause for 1 second.

sleep 1;

# Printing '|' and moving the cursor back.

echo -ne '\b|';

# Another pause for 1 second.

sleep 1;

# Printing '/' and moving the cursor back.

echo -ne '\b/';

# Another pause for 1 second.

sleep 1;

# Printing '-' and moving the cursor back.

echo -ne '\b-';

# Another pause for 1 second.

sleep 1;

# Printing '\' and moving the cursor back.

echo -ne '\b\\';

sleep 1;

# Another pause for 1 second.

# Printing '|' and moving the cursor back.

echo -ne '\b|';

# Printing a backspace to move the cursor back and overwrite the last character, followed by a newline to move to the next line.

echo -ne "\b \n";

2. Modify the code to make the spinner spin the other direction.

#!/bin/bash

echo -ne '\\';

sleep 1;

echo -ne '\b-';

sleep 1;

echo -ne '\b/';

sleep 1;

echo -ne '\b|';

sleep 1;

echo -ne '\b\\';

sleep 1;

echo -ne '\b-';

sleep 1;

echo -ne '\b/';

sleep 1;

echo -ne '\b|';

echo -ne "\b \n";

3.Modify the code to make the spinner spin faster.

#!/bin/bash

echo -ne '\\';

sleep 0.5;

echo -ne '\b-';

sleep 0.5;

echo -ne '\b/';

sleep 0.5;

echo -ne '\b|';

sleep 0.5;

echo -ne '\b\\';

sleep 0.5;

echo -ne '\b-';

sleep 0.5;

echo -ne '\b/';

sleep 0.5;

echo -ne '\b|';

echo -ne "\b \n";

4.Modify the code so that there are 10 revolutions of the spinner using a for

#!/bin/bash

# Number of revolutions

num\_revolutions=10

for ((i=0; i<num\_revolutions; i++)); do

echo -ne '\\';

sleep 0.5;

echo -ne '\b-';

sleep 0.5;

echo -ne '\b/';

sleep 0.5;

echo -ne '\b|';

sleep 0.5;

echo -ne '\b\\';

sleep 0.5;

echo -ne '\b-';

sleep 0.5;

echo -ne '\b/';

sleep 0.5;

echo -ne '\b|';

done

echo -ne "\b \n";

5.Be creative and make a variety of spinning shapes that include both forward and backward spin action such that Bran, your TA, is impressed and amazed when he runs it. Make sure your spinning designs have at least 10 revolutions using the for loop, and that your code is well commented. 6.Upload this modified bash script to your Canvas assignments site for this class.

#!/bin/bash

# Number of revolutions

num\_revolutions=10

# Function to display a forward spinner

function forward\_spinner {

echo -ne '\\';

sleep 0.2;

echo -ne '\b-';

sleep 0.2;

echo -ne '\b/';

sleep 0.2;

echo -ne '\b|';

sleep 0.2;

}

# Function to display a backward spinner

function backward\_spinner {

echo -ne '/';

sleep 0.2;

echo -ne '\b-';

sleep 0.2;

echo -ne '\b\\';

sleep 0.2;

echo -ne '\b|';

sleep 0.2;

}

# Loop for 10 revolutions

for ((i=0; i<num\_revolutions; i++)); do

# Display forward spinner

forward\_spinner

# Display backward spinner

backward\_spinner

done

# Clear the spinner and move to the next line

echo -ne "\b \n";

**Problem 2 (5 pts)**

Write a well-commented bash script that outputs each line of the file you previously downloaded (in Lecture 3) called hapmap1.ped one at a time, waiting 1 second between the display of each line. You should use the following commands in your script: wc, basename, for, sleep, tail, and seq. When you run the script, it should look something like the Matrix, but with genotypes instead of binary 1s and 0s. Name your script thegeneticsmatrix.sh and upload that script to your Canvas assignments page.

#!/bin/bash

# Get the total number of lines in the file

total\_lines=$(wc -l < hapmap1.ped)

# Extract the filename without the path

filename=$(basename hapmap1.ped)

# Loop through line numbers using seq

for line\_number in $(seq 1 $total\_lines); do

# Extract the line from the file using tail

line=$(tail -n +$line\_number hapmap1.ped | head -n 1)

# Convert binary genotypes to actual genotypes

converted\_line=$(echo "$line" | sed 's/0/PAX6/g; s/1/NESTIN/g; s/2/CTIP2/g')

# Print the line with the filename and genotypes

echo "$filename: $converted\_line"

# Waiting 1 second between the display of each line

sleep 1

done

**Problem 3 (5 pts)**

Make one bash script to generate a series of files using two nested for loops. These will be data files representing 50 donors each of which has 10 time points. The files should be named like donor1\_tp1.txt, donor1\_tp2.txt, etc. In total there should be 500 files that your script creates. Within each file there should be a header row that says “data” and then there should be 5 random numbers in a column. You can use $RANDOM to create random numbers in bash.

#!/bin/bash

# Define the number of donors and time points

num\_donors=50

num\_time\_points=10

# Create files for each donor and time point

for ((donor=1; donor<=num\_donors; donor++)); do

for ((tp=1; tp<=num\_time\_points; tp++)); do

# Define the filename

filename="donor${donor}\_tp${tp}.txt"

# Create the file and write the header

echo "data" > "$filename"

# Write 5 random numbers to the file

for ((i=1; i<=5; i++)); do

echo "$RANDOM" >> "$filename"

done

echo "Created file: $filename"

done

done

**Problem 4 (5 pts)**

You just created a whole lot of messy files, write a bash script to organize them called organizedata.sh. Put each donor’s 10 files into their own directory. Put every donor’s file into a fakedata directory. Make all files read only so that they cannot be accidentally edited. Upload your well commented script to your Canvas assignments.

#!/bin/bash

# Create the fakedata directory

mkdir -p fakedata

# Loop through each donor file

for donor\_file in donor\*\_tp\*.txt; do

# Extract donor number from the filename

donor\_number=$(echo "$donor\_file" | cut -d "\_" -f 1 | sed 's/donor//')

# Create a directory for the donor

donor\_dir="fakedata/donor${donor\_number}"

mkdir -p "$donor\_dir"

# Move the file to the donor's directory

mv "$donor\_file" "$donor\_dir/"

# Set the file as read-only

chmod -w "$donor\_dir/$donor\_file"

done

echo "Data organized!"