# Array of options for the user to choose from, based on problem description

options=("Create a file" "Create a directory" "Delete a file" "Delete a directory" "Count from/to n" "Search for a phrase in a given directory" "Exit")

# Stores the requested user action number

action=0

# Read the user's name; -r to read entire line

echo "Please enter your name:"

read -r name

# Read the user's ID

echo "Please enter your ID:"

read id

# Check if the user ID is a valid integer using regex;

# and check if user ID is even;

# if both checks do not pass, exit.

if ! { [[ "$id" =~ ^[0-9]+$ ]] && ((id % 2 == 0)); }; then

echo -e "\nID not even, goodbye $name!"

exit 0

fi

# Display greeting; -e to allow \n

echo -e "\nGood morning $name, the date and time is $(date +"%a %b %d %I:%M:%S %p%n%Z %Y"), how can I help you today?"

# Displays the given text as colored, then returns to white text.

# First parameter is the text to be displayed.

# Second parameter is the special color character.

display\_colored() {

# Data

local txt=$1

# Color

local color=$2

# Display info

echo -en "$color$txt$(tput setaf 7)"

}

# Displays an error message (red).

# Takes text as input.

display\_err() {

# Display colored information

local err=$1

# Display colored information (red)

display\_colored "\n$err\n" "$(tput setaf 1)"

}

# Displays a success message, colored green.

# Takes text as input.

display\_success() {

# Input message

local msg=$1

# True removes new line characters

local rmv\_pad=${2:-false}

if [ $rmv\_pad != true ]; then

msg="\n$msg\n"

fi

# Display colored information (green).

display\_colored "$msg" "$(tput setaf 2)"

}

# Displays an informational message, colored cyan.

# Takes text as input.

display\_info() {

# Input message

local info=$1

# Display colored information (cyan).

display\_colored "\n$info\n" "$(tput setaf 6)"

}

# Function to display the menu & set the chosen user action

display\_menu() {

# Empty new line

echo ""

# Iterate over the length of the options array

for ((i = 0; i < "${#options[@]}"; i++)); do

# Print the option

echo "$((i + 1))- ${options[i]}"

done

# Empty new line

echo ""

# Read user choice from console

read action

# Check if the user input is a valid integer using regex;

# if check does not pass, set the choice to -1.

if ! { [[ "$action" =~ ^[0-9]+$ ]]; }; then

action=-1

fi

}

# Transforms the given path from relative to absolute.

# Takes path to be transformed as first input.

to\_absolute() {

# Prepend the home-dir path to the given path.

echo "$HOME/$1"

}

# Performs the necessary I/O operations to create a new file.

# Treats the given path as absolute.

# Takes the file path as the first parameter.

# Takes an optional boolean as the second parameter,

# if true create a dir; otherwise a file.

create\_file() {

# Absolute path

local path=$(to\_absolute "$1")

# Is directory flag, default is false.

local is\_dir=${2:-false}

# If directory

if [ $is\_dir == true ]; then

# Check if directory does not exist, or is a file name

if [[ ! -d "$path" ]] && [[ ! -f "$path" ]]; then

# -p to create missing intermediate dirs

mkdir -p "$path"

display\_success "Directory created successfully!"

else # Does exist

display\_err "Directory already exists!"

fi

else # If file

# Check if file does not exist, or is a directory name

if [[ ! -f "$path" ]] && [[ ! -d "$path" ]]; then

# First create intermediates if missing

mkdir -p "${path%/\*}" && touch "$path"

display\_success "File created successfully!"

else # Does exist

display\_err "File already exists!"

fi

fi

}

# Performs the necessary I/O operations to delete a file.

# Treats the given path as absolute.

# Takes the file path as the first parameter.

# Takes an optional boolean as the second parameter,

# if true delete a dir; otherwise a file.

delete\_file() {

# Absolute path

local path=$(to\_absolute "$1")

# Is directory flag, default is false.

local is\_dir=${2:-false}

# If directory

if [ $is\_dir == true ]; then

# Check if directory does not exist

if [ ! -d "$path" ]; then

display\_err "Directory does not exist!"

else # Does exist

rm -ir "$path" # Ask for confirmation; -i

display\_success "Directory deleted successfully!"

fi

else # If file

# Check if file does not exist

if [ ! -f "$path" ]; then

display\_err "File does not exist!"

else # Does exist

rm -i "$path" # Ask for confirmation; -i

display\_success "File deleted successfully!"

fi

fi

}

# Displays the numbers from lower to upper; [lower, upper].

# Takes lower bound as the first parameter.

# Takes upper bound as the second parameter.

# Note that both bounds are valid when given to the function.

count() {

# Positive integer input

local lower=$1

# Positive integer input

local upper=$2

# Loop over the numbers in [lower, upper]

for ((i = lower; i < upper; i++)); do

display\_success "$i, " true

done

display\_success "$upper\n" true

}

# Searches for the given phrase in the given directory.

# Takes phrase as first parameter.

# Takes absolute path to directory as second parameter.

search() {

# Phrase to look for in the directory files

local phrase=$1

# Absolute path to directory

local path=$(to\_absolute "$2")

# Check if directory does not exist

if [ ! -d "$path" ]; then

display\_err "Directory does not exist!"

else # Does exist

# Display the search result

# Search recursively (-r), add line number (-n), & match the entire phrase (-w).

# -e for the search pattern (phrase).

# Store result for display

local result=$(grep -rnw "$path" -e "$phrase")

# Check if there are no results

if [ -z "$result" ]; then

display\_info "Phrase not found!"

else

display\_info "$result"

fi

display\_success "Search complete!"

fi

}

# Keep on taking user input till exiting

while true; do

# Call the display menu, fill take action input

display\_menu

# Switch user action

case $action in

1) # Create file

# Take file path

display\_info "Enter the file’s absolute path: "

read -r file\_path

create\_file "$file\_path"

;;

2) # Create dir

# Take dir path

display\_info "Enter the directory's absolute path: "

read -r dir\_path

create\_file "$dir\_path" true

;;

3) # Delete file

# Take file path

display\_info "Enter the file’s absolute path: "

read -r file\_path

delete\_file "$file\_path"

;;

4) # Delete directory

# Take dir path

display\_info "Enter the directory's absolute path: "

read -r dir\_path

delete\_file "$dir\_path" true

;;

5) # Count from 1 to n

# Take lower bound

display\_info "Enter lower bound (positive integer): "

read lower

# Take upper bound

display\_info "Enter upper bound (positive integer): "

read upper

# Check if the user input is a valid integer using regex;

# if check does not pass, warn user.

if ! { [[ "$lower" =~ ^[-+]?[0-9]+$ ]]; }; then

display\_err "Invalid input, must be an integer!"

else # valid lower bound

# Check if the user input is a valid integer using regex;

# and check if user lower bound is less than or equal to the upper;

# if both checks do not pass, warn user.

if ! { [[ "$upper" =~ ^[-+]?[0-9]+$ ]] && ((lower <= upper)); }; then

display\_err "Invalid input, must be greater than lower bound!"

else # valid integer

count "$lower" "$upper"

fi

count "$n"

fi

;;

6) # Search phrase

# Take phrase to look for

display\_info "Enter the phrase to look for: "

read -r search\_phrase

# Take directory to look in

display\_info "Enter the directory's absolute path: "

read -r search\_dir

search "$search\_phrase" "$search\_dir"

;;

7) # Exit

display\_info "Exiting the script. Goodbye!"

exit 0

;;

\*)

display\_err "Invalid choice. Please enter a number between 1 and ${#options[@]}."

;;

esac

done