LAB-1

```
# Check if three arguments are passed
if [ "$#" -ne 3 ]; then
  echo "Usage: $0 num1 num2 num3"
  exit 1
fi
# Read command-line arguments
num1=$1
num2=$2
num3=$3
# Find the greatest number
if [ "$num1" -ge "$num2" ] && [ "$num1" -ge "$num3" ]; then
  echo "The greatest number is: $num1"
elif [ "$num2" -ge "$num1" ] && [ "$num2" -ge "$num3" ]; then
  echo "The greatest number is: $num2"
else
  echo "The greatest number is: $num3"
fi
```

LAB2

```
if [ $# -eq 0 ]; then
        echo "Usage: #0 num"
```

```
    exit 1
fi
num1=$1
if [ $((num1%2)) -eq 0 ]; then
    echo "$num1 is even"
else
    echo "$num1 is odd"
fi
```

LAB3

```
if [ $# -eq 0 ]; then
    echo "Usage: $0 num1 num2 ... numn"
    exit 1
fi
sum=0
for num in "$@"; do
    sum=$((sum + num))
done
average=$((sum / $#))
echo "The average is : $average"
```

LAB4

```
# Function to check if a number is prime
is_prime() {
    local num=$1

    # Check if the number is less than 2
    if [ $num -le 1 ]; then
        echo "$num is not prime"
        return
    fi

    # Check for divisibility from 2 to the square root of num
    for ((i=2; i*i<=num; i++)); do
        if [ $((num % i)) -eq 0 ]; then
            echo "$num is not prime"
            return
        fi
    done

    # If no divisors were found, it's prime
    echo "$num is prime"
}

# Check if the script was passed a number
if [ $# -eq 0 ]; then
```

```
    echo "Usage: $0 <number>"
    exit 1
fi

# Call the is_prime function with the provided number
is_prime $1
```

LAB5

```
# Read the input from the user
read -p "Enter the input: " input

# Check if the input is a number (integer or floating point)
if [[ "$input" =~ ^-?[0-9]+(\.[0-9]+)?$ ]]; then
    echo "The input is a number."
else
    echo "The input is a string."
fi
```

LAB6

```
# Check if a filename is provided as an argument
if [ "$#" -ne 1 ]; then
    echo "Usage: $0 <filename>"
    exit 1
```

```
fi
filename=$1
# Check if the file exists
if [ ! -f "$filename" ]; then
   echo "File not found!"
   exit 1
fi

# Process each line of the file
line_number=1
while IFS= read -r line; do
   char_count=$(echo -n "$line" | wc -c)
   word_count=$(echo "$line" | wc -w)
   echo "Line $line_number: Characters = $char_count, Words = $word_count"
   ((line_number++))
done < "$filename"
```

LAB7

```
# Check if the user provided an argument
if [ $# -ne 1 ]; then
   echo "Usage: $0 <number_of_terms>"
   exit 1
fi
```

```
# Read the number of terms
n=$1
# Check if the input is a valid positive integer
if ! [[ "$n" =~ ^[0-9]+$ ]] || [ "$n" -le 0 ]; then
    echo "Please enter a positive integer."
    exit 1
fi
# Initialize the first two terms
a=0
b=1
echo "Fibonacci series up to $n terms:"
# Generate Fibonacci series
for (( i=1; i<=n; i++ )); do
    echo -n "$a "
    # Calculate the next term
    next=$((a + b))
    # Update terms
    a=$b
    b=$next
done

echo
```

LAB8

```
if [ $# -ne 1 ]; then
```

```
    echo "Usage: $0 <number>"
    exit 1
fi
# Read the input number
num=$1

# Validate that the input is a non-negative integer
if ! [[ "$num" =~ ^[0-9]+$ ]]; then
    echo "Please enter a non-negative integer."
    exit 1
fi
# Initialize factorial
factorial=1
# Calculate factorial using a loop
for (( i=1; i<=num; i++ )); do
    factorial=$((factorial * i))
done
# Print the result
echo "Factorial of $num is $factorial"
```

LAB9

```
# Check if a number is provided
```

```
if [ $# -ne 1 ]; then
    echo "Usage: $0 <number>"
    exit 1
fi

# Read the input number
num=$1

# Validate that the input is a positive integer
if ! [[ "$num" =~ ^[0-9]+$ ]]; then
    echo "Please enter a non-negative integer."
    exit 1
fi
# Initialize sum to 0
sum=0

# Loop to calculate the sum of digits
while [ "$num" -gt 0 ]; do
    digit=$((num % 10))  # Extract the last digit
    sum=$((sum + digit)) # Add the digit to the sum
    num=$((num / 10))    # Remove the last digit
done
# Print the result
echo "Sum of digits is $sum"
```

LAB10

```
# Check if a string is provided
if [ $# -ne 1 ]; then
   echo "Usage: $0 <string>"
   exit 1
fi
input=$1

# Reverse the string
reversed=$(echo "$input" | rev)

# Check if the original string and reversed string are the same
if [ "$input" = "$reversed" ]; then
   echo "The string '$input' is a palindrome."
else
   echo "The string '$input' is not a palindrome."
fi
```

LAB11

```
# Prompt the user to enter a string
echo "Enter a string:"
read input
```

```
# Check the length of the string
if [ ${#input} -lt 5 ]; then
   echo "The string does not have at least 5 characters."
else
   echo "The string has at least 5 characters."
fi
```

LAB12

```
# Check if an argument is provided
if [ $# -ne 1 ]; then
   echo "Usage: $0 <string>"
   exit 1
fi

# Read the input string from the argument
input=$1

# Calculate the length of the string
length=${#input}

echo "The length of the string '$input' is $length."
```

LAB13

```
if [ "$#" -ne 2 ]; then
  echo "Usage: $0 <directory1> <directory2>"
```

```
  exit 1
fi



# Assign arguments to variables
DIR1=$1
DIR2=$2

if [ ! -d "$DIR1" ]; then
  echo "Error: $DIR1 is not a directory."
  exit 1
fi
if [ ! -d "$DIR2" ]; then
  echo "Error: $DIR2 is not a directory."
  exit 1
fi

for file1 in "$DIR1"/*; do
  filename=$(basename "$file1")

  # Check if a file with the same name exists in DIR2
  if [ -f "$DIR2/$filename" ]; then
    # Compare files, and if they are identical, delete the one in DIR1
    if cmp -s "$file1" "$DIR2/$filename"; then
```

```
      echo "Deleting $file1 (matches $DIR2/$filename)"
      rm "$file1"
    fi
  fi
done
echo "Matching files deleted from $DIR1."
```

LAB14

```
# Loop to run 3 times
for ((i=1; i<=3; i++)); do
    echo "Iteration $i: Processes running on the system at $(date):"
    ps -e  # Display all running processes
    echo "--------------------------------------"

    # Pause for 30 seconds after each iteration, except the last one
    if [ $i -lt 3 ]; then
        sleep 30
    fi
done
```

LAB15

```
# Check if a filename is provided as an argument
if [ $# -ne 1 ]; then
```

```
    echo "Usage: $0 <filename>"
    exit 1
fi
# Get the filename from the argument
filename=$1

# Check if the file exists
if [ ! -f "$filename" ]; then
    echo "Error: File '$filename' does not exist."
    exit 1
fi

# Display the last modification time of the file
mod_time=$(stat -c %y "$filename")
echo "The last modification time of the file '$filename' is: $mod_time"
```

LAB16

```
# Check if a filename is provided as an argument
if [ $# -ne 1 ]; then
    echo "Usage: $0 <filename>"
    exit 1
fi

# Get the filename from the argument
```

```
filename=$1

# Check if the file exists
if [ ! -f "$filename" ]; then
    echo "Error: File '$filename' does not exist."
    exit 1
fi

# Check spelling using aspell
echo "Checking spelling in the file '$filename':"
aspell list < "$filename"
```

LAB17

```
echo -n "Enter a file name: "
read file

if [ ! -f "$file" ]; then
    echo "$file not a file!"
    exit 1
fi

echo -n "Enter a Password: "
read -s password
echo
```

```
# Use openssl for encryption with -pbkdf2
openssl enc -aes-256-cbc -salt -pbkdf2 -in "$file" -out "$file.cpy" -k "$password"

if [ $? -eq 0 ]; then
   echo "$file.cpy created as encrypted file"
else
   echo "Error: Encryption failed"
   exit 2
fi
```

LAB18

```
# Check if a filename argument is provided
if [ $# -ne 1 ]; then
   echo "Usage: $0 <filename>"
   exit 1
fi

# Assign the filename argument to a variable
input_file="$1"

# Check if the file exists
if [ ! -f "$input_file" ]; then
   echo "File not found: $input_file"
   exit 2
fi
```

```
# Create a temporary file securely using mktemp
temp_file=$(mktemp)

# Convert the content to lowercase, remove non-alphanumeric characters, and extract words
tr '[:upper:]' '[:lower:]' < "$input_file" | \
tr -c '[:alnum:]' '[\n*]' | \
sort | \
uniq > "$temp_file"

# Move the processed wordlist to a new file named "wordlist.txt"
mv "$temp_file" wordlist.txt

echo "Wordlist has been extracted and saved to 'wordlist.txt'."
```

LAB19

```
# Check if a filename is provided as an argument
if [ $# -ne 2 ]; then
    echo "Usage: $0 <input-file> <output-file>"
    exit 1
fi

# Get the input and output filenames
input_file=$1
output_file=$2
```

```
# Check if the input file exists
if [ ! -f "$input_file" ]; then
    echo "Error: File '$input_file' does not exist."
    exit 1
fi

# Remove all blank spaces (spaces, tabs, and newlines) and redirect the output
tr -d '[:space:]' < "$input_file" > "$output_file"

# Confirm successful processing
echo "All blank spaces have been removed. Output saved to '$output_file'."
```

LAB20

```
# Check if a filename is provided as an argument
if [ $# -ne 1 ]; then
    echo "Usage: $0 <filename>"
    exit 1
fi

# Get the original filename
original_filename=$1
if [ ! -f "$original_filename" ]; then
    echo "Error: File '$original_filename' does not exist."
```

```
    exit 1
fi

# Convert the filename to lowercase
lowercase_filename=$(echo "$original_filename" | tr '[:upper:]' '[:lower:]')

# Check if the lowercase name is different
if [ "$original_filename" = "$lowercase_filename" ]; then
    echo "The filename is already in lowercase: $original_filename"
    exit 0
fi

# Rename the file
mv "$original_filename" "$lowercase_filename"

# Confirm the renaming
echo "The file has been renamed to: $lowercase_filename"
```

LAB21

```
# Check if a filename is provided as an argument
if [ $# -ne 1 ]; then
    echo "Usage: $0 <filename>"
    exit 1
fi
```

```
# Get the filename
filename=$1

# Check if the file exists
if [ ! -f "$filename" ]; then
    echo "Error: File '$filename' does not exist."
    exit 1
fi

# Convert the content to lowercase and overwrite the file
temp_file=$(mktemp)
tr '[:upper:]' '[:lower:]' < "$filename" > "$temp_file" && mv "$temp_file"
"$filename"

# Confirm the conversion
if [ $? -eq 0 ]; then
    echo "All characters in '$filename' have been converted to lowercase."
else
    echo "Error: Failed to process the file."
fi
```

LAB22

```
# Check if exactly 3 files are provided as arguments
if [ $# -ne 3 ]; then
```

```
    echo "Usage: $0 <file1> <file2> <file3>"
    exit 1
fi

# Get the filenames from the arguments
file1=$1
file2=$2
file3=$3

# Check if all the files exist
if [ ! -f "$file1" ]; then
    echo "Error: File '$file1' does not exist."
    exit 1
fi

if [ ! -f "$file2" ]; then
    echo "Error: File '$file2' does not exist."
    exit 1
fi

if [ ! -f "$file3" ]; then
    echo "Error: File '$file3' does not exist."
    exit 1
fi
```

```
# Combine the files into a new file (e.g., combined.txt)
combined_file="combined.txt"
cat "$file1" "$file2" "$file3" > "$combined_file"

# Display the word count of the combined file
word_count=$(wc -w < "$combined_file")
echo "The word count of the combined file is: $word_count"
```

LAB23

```
# Check if a filename is provided as an argument
if [ $# -ne 1 ]; then
    echo "Usage: $0 <filename>"
    exit 1
fi
filename=$1
# Check if the file exists
if [ ! -f "$filename" ]; then
    echo "Error: File '$filename' does not exist."
    exit 1
fi

# Initialize counters for line numbers
line_number=1
```

```
# Loop through each line of the file
while IFS= read -r line
do
    # Check if the line number is odd or even
    if (( line_number % 2 == 0 )); then
        # Write even-numbered lines to evenfile
        echo "$line" >> evenfile
    else
        # Write odd-numbered lines to oddfile
        echo "$line" >> oddfile
    fi
    # Increment line number
    ((line_number++))
done < "$filename"
```

LAB24

```
# Check if the filename is provided as an argument
if [ $# -ne 1 ]; then
    echo "Usage: $0 <filename>"
    exit 1
fi

# Get the input filename
```

```
filename=$1
if [ ! -f "$filename" ]; then
    echo "Error: File '$filename' does not exist."
    exit 1
fi

# Initialize line number counter
line_number=1

# Temporary file to store the modified content
temp_file=$(mktemp)

# Loop through the lines of the file
while IFS= read -r line; do
    # If the line number is odd, write it to the temporary file
    if ((line_number % 2 != 0)); then
        echo "$line" >> "$temp_file"
    fi
    # Increment the line number
    ((line_number++))
done < "$filename"

# Move the modified content back to the original file
mv "$temp_file" "$filename"
```

LAB25

```
# Get the username
username=$(whoami)

# Get the current date and time
current_datetime=$(date)

# Get the list of users currently logged in
logged_in_users=$(who)

# Output the username
echo "Username: $username"
echo "*******"

# Output the current date and time
echo "Current date and time: $current_datetime"
echo "*******"

# Output the users who are logged in
echo "Users logged in:"
echo "$logged_in_users"
echo "*******"
```