Green University of Bangladesh
Department of Computer Science and Engineering(CSE)

Faculty of Sciences and Engineering

Semester: (Spring , Year:2024), B.Sc. in CSE (Day)

LAB REPORT NO #02

Course Title: Operating System Lab

Course Code: CSE - 310 **Section: 213_D5**

Lab Experiment Name: Shell Scripting- I & II.

<u>Student Details</u>

Name	ID
MD Dulal Hossain	213902116

Lab Date : 28 - 03 - 2024
Submission Date : 26 - 04 - 2023

Course Teacher's Name : Md. Solaiman Mia

[For Teachers use only: Don't Write Anything inside this box]

<u>Lab Report Status</u>

Marks: …… **Signature:.....**
Comments:..... **Date:.....**

Problem 1

Title :

Write a Shell program to find the sum of odd and even numbers from a set of numbers.

Algorithms :

1. Initialize sum_odd and sum_even variables to 0.
2. Loop through each number passed as input:
3. Check if the number is even:
4. If yes, add it to the sum_even variable.
5. If no , add it to the sum_odd variable.
6. Print the sum of even numbers.
7. Print the sum of odd numbers.

Code :

```
dulal@213902116: ~/Documents/cse-310/lab_report_2
  GNU nano 6.2                          problem1.sh *
#!/bin/bash
sum_odd=0
sum_even=0
for num in "$@"
do
  if [ $((num % 2)) -eq 0 ]; then
    sum_even=$((sum_even + num))
  else
    sum_odd=$((sum_odd + num))
  fi
done
echo "Sum of even numbers: $sum_even"
echo "Sum of odd numbers: $sum_odd"
```

Figure 1.1 : Code for problem 1.

Output :

```
dulal@213902116: ~/Documents/cse-310...
dulal@213902116:~/Documents/cse-310/lab_report_2$ ./problem1.sh
6 4 5 7 6 4 2 13
Sum of even numbers: 22
Sum of odd numbers: 25
dulal@213902116:~/Documents/cse-310/lab_report_2$
```

Figure 1.2 : Output in show Successfully for problem 1.

Problem 2

Title :

Write a Shell program to Check Triangle is Valid or Not.

Algorithms :

1. Define a function is_valid_triangle to check if three given numbers can form a triangle.
2. Check the number of arguments: If the number of arguments is not exactly three, print an error message and exit.
3. Call the function is_valid_triangle with the three numbers as arguments.
4. In the function is_valid_triangle:
   Check if the sum of any two sides is greater than the third side for all combinations.
   If yes, print that the sides form a valid triangle.
   If not, print that the sides do not form a valid triangle.

Code :

```
dulal@213902116: ~/Documents/cse-310...
  GNU nano 6.2                    problem2.sh *
#!/bin/bash
is_valid_triangle() {
  if [ $(($1 + $2)) -gt $3 ] && [ $(($2 + $3)) -gt $1 ] && [ $(>
    echo "The sides $1, $2, and $3 form a valid triangle."
  else
    echo "The sides $1, $2, and $3 do not form a valid triangle>
  fi
}
if [ $# -ne 3 ]; then
  echo "Please provide exactly three numbers as sides of a tria>
  exit 1
fi
is_valid_triangle $1 $2 $3
```

Figure 2.1 : Code for problem 2.

Output :

```
dulal@213902116: ~/Documents/cse-310/lab_rep...
dulal@213902116:~/Documents/cse-310/lab_report_2$ ./problem2.sh 3 4 5
The sides 3, 4, and 5 form a valid triangle.
dulal@213902116:~/Documents/cse-310/lab_report_2$ ./problem2.sh 1 2 20
The sides 1, 2, and 20 do not form a valid triangle.
dulal@213902116:~/Documents/cse-310/lab_report_2$
```

Figure 2.2 : Output in show Successfully for problem 2.

Problem 3

Title :

Write a shell program to display odd position numbers (using For loop).
Sample Input: Enter 7-digit number: 5867458
Output: 5 6 4 8

Algorithms :

1. Prompt the user to enter a 7-digit number.
2. Validate the input: Check if the entered value consists of exactly 7 digits using a regular expression. If not, print an error message and exit.
3. Iterate through each digit of the entered number: For each digit, determine its position in the number (starting from 0). Check if the position is odd:
   If the position is odd, print the digit.

Code :

```
dulal@213902116: ~/Documents/cse-310/lab_report_2
  GNU nano 6.2                    problem3.sh *
#!/bin/bash
read -p "Enter 7-digit number: " number
if [[ ! $number =~ ^[0-9]{7}$ ]]; then
  echo "Please enter a valid 7-digit number."
  exit 1
fi
for (( i=0; i<${#number}; i++ )); do
  digit=${number:$i:1}
  # Check if the position is odd
  if (( $((i % 2)) == 0 )); then
    echo $digit
  fi
done
```

Figure 3.1 : Code for problem 3.

Output :

```
dulal@213902116: ~/Documents/cse-310/lab_report_2
dulal@213902116:~/Documents/cse-310/lab_report_2$ ./problem3.sh
Enter 7-digit number: 1234567
1
3
5
7
dulal@213902116:~/Documents/cse-310/lab_report_2$
```

Figure 3.2 : Output in show Successfully for problem 3.

Problem 4

Title :

• Write a Shell program using while loop: Sample Input: Enter the number: 148541547854 Output: 1 = 2 times , 4 = 4 times, 8 = 2 times, 5 = 3 times, 7 = 1 times

Algorithms :

1. Prompt the user to enter a number.
2. Declare an associative array digit_count to store the count of each digit.
3. Determine the length of the entered number.
4. Iterate through each digit of the entered number: Extract each digit from the number using substring extraction. Increment the count of the corresponding digit in the digit_count array.
5. Print the counts of each digit: Iterate through the keys of the digit_count array. Print each digit along with its count.

Code :

```
dulal@213902116: ~/Documents/cse-310/lab_report_2
  GNU nano 6.2                    problem4.sh *
#!/bin/bash
read -p "Enter the number: " input_number
declare -A digit_count
number_length=${#input_number}
index=0
while [ $index -lt $number_length ]; do
  digit=${input_number:$index:1}
  ((digit_count[$digit]++))
  ((index++))
done
for digit in "${!digit_count[@]}"; do
  echo "$digit = ${digit_count[$digit]} times"
done
```

Figure 4.1 : Code for problem 4.

Output :

```
dulal@213902116: ~/Documents/cse-310/lab_report_2
dulal@213902116:~/Documents/cse-310/lab_report_2$ ./problem4.sh
Enter the number: 123231214131213
4 = 1 times
3 = 4 times
2 = 4 times
1 = 6 times
dulal@213902116:~/Documents/cse-310/lab_report_2$
```

Figure 4.2 : Output in show Successfully for problem 4.

Problem 5

Title :

Write a Shell program to find the 2nd highest and 3rd highest numbers from a set of numbers and sum of them using array.

Sample Input: Enter the number of elements: 5

Enter the number: 10, Enter the number: 21, Enter the number: 30

Enter the number: 17, Enter the number: 5

Output: The sum of first and last element is: (21+17) = 38

Algorithms :

1. Prompt the user to enter the number of elements.
2. Declare an array named numbers to store the elements.
3. Read numbers into the array using a loop.
4. Sort the array in descending order: Convert the array elements to a newline-separated list. Sortlist numerically in reverse order.Convert sorted list back to a space-separated string.
5. Extract the 2nd and 3rd highest numbers from the sorted array.
6. Calculate the sum of the 2nd and 3rd highest numbers.
7. Print the result: Print the sum of the 2nd and 3rd highest elements.

Code :

```
dulal@213902116: ~/Documents/cse-310/lab_report_2
  GNU nano 6.2                    problem5.sh
echo "Enter the number of elements: "
read n
declare -a numbers
for (( i=0; i<n; i++ )); do
  echo "Enter the number: "
  read numbers[$i]
done
sorted_numbers=($(echo "${numbers[@]}" | tr ' ' '\n' | sort -nr | tr '\n
second_highest=${sorted_numbers[1]}
third_highest=${sorted_numbers[2]}
sum=$((second_highest + third_highest))
echo "The sum of the 2nd and 3rd highest elements is: ($second_highest+$
```

Figure 5.1 : Code for problem 5.

Output :

```
dulal@213902116: ~/Documents/cse-310/lab_report_2
dulal@213902116:~/Documents/cse-310/lab_report_2$ ./problem5.sh
Enter the number of elements:
3
Enter the number:
12
Enter the number:
23
Enter the number:
9
The sum of the 2nd and 3rd highest elements is: (12+9) = 21
dulal@213902116:~/Documents/cse-310/lab_report_2$
```

Figure 5.2 : Output in show Successfully for problem 5.

Problem 6

Title :

Write a Shell program to find factorial of two different num & sum of number using function.
Sample Input: 5, 6
Output: Factorial of 5 is 120 Factorial of 6 is 720, 120 + 720 = 840

Algorithms :

1. Define a function named factorial to calculate factorial of a given number: Take number as input. Initialize a variable fact to store the factorial and set it to 1. Use a loop to multiply fact by each number from 2 to the given number. Return the calculated factorial.
2. Prompt the user to enter two numbers.
3. Calculate the factorial of each entered number using the factorial function.
4. Calculate the sum of the factorials.
5. Print the results: Print the factorial of each entered number. Print sum of the factorials.

Code :

```
dulal@213902116: ~/Documents/cse-310/lab_report_2
  GNU nano 6.2                    problem6.sh
factorial() {
  local num=$1
  local fact=1
  for (( i=2; i<=num; i++ )); do
    fact=$((fact * i))
  done
  echo $fact
}
read -p "Enter the first number: " num1
read -p "Enter the second number: " num2
fact1=$(factorial $num1)
fact2=$(factorial $num2)
sum=$((fact1 + fact2))
echo "Factorial of $num1 is $fact1"
echo "Factorial of $num2 is $fact2"
echo "$fact1 + $fact2 = $sum"
```

Figure 6.1 : Code for problem 6.

Output :

```
dulal@213902116: ~/Documents/cse-310/lab_report_2
dulal@213902116:~/Documents/cse-310/lab_report_2$ ./problem6.sh
Enter the first number: 5
Enter the second number: 6
Factorial of 5 is 120
Factorial of 6 is 720
120 + 720 = 840
dulal@213902116:~/Documents/cse-310/lab_report_2$
```

Figure 6.2 : Output in show Successfully for problem 6.

Problem 7

Title :

Write a Shell program to find total number of alphabets, digits or special characters in a string.
Sample Input: Today is 12 November.
Output: ,Alphabets = 15 , Digits = 2 & Special characters = 4

Algorithms :

1. Prompt the user to enter a string.
2. Initialize three variables count_alpha, count_digit, and count_special to store the counts of alphabets, digits, and special characters, respectively.
3. Iterate through each character in the input string: Extract each character from the string. Check if the character is an alphabet using a regular expression. If it is, increment the count_alpha variable. Check if the character is a digit using a regular expression.
   If it is, increment the count_digit variable.
   If the character is neither an alphabet nor a digit, increment the count_special variable.
4. Print the counts of alphabets, digits, and special characters.
5. That's it! The script calculates and prints the counts of alphabets, digits, and special characters in the input string.

Code :

```
dulal@213902116: ~/Documents/cse-310/lab_report_2
  GNU nano 6.2                    problem7.sh
read -p "Enter the string: " input_string
count_alpha=0
count_digit=0
count_special=0
for (( i=0; i<${#input_string}; i++ )); do
  char=${input_string:$i:1}
  if [[ $char =~ [A-Za-z] ]]; then
    ((count_alpha++))
  elif [[ $char =~ [0-9] ]]; then
    ((count_digit++))
  else
    ((count_special++))
  fi
done
echo "Alphabets = $count_alpha"
echo "Digits = $count_digit"
echo "Special characters = $count_special"
```

Figure 7.1 : Code for problem 7.

Output :

```
dulal@213902116: ~/Documents/cse-310/lab_report_2
dulal@213902116:~/Documents/cse-310/lab_report_2$ ./problem7.sh
Enter the string: my name is dulal, id: 213902116.
Alphabets = 15
Digits = 9
Special characters = 8
dulal@213902116:~/Documents/cse-310/lab_report_2$
```

Figure 7.2 : Output in show Successfully for problem 7.