SHELL SCRIPTING

Previous Year Practical Question 2020

1. Write a shell script to find first N prime numbers.

Answer:

Shell Script

```bash
#!/bin/bash
echo "Enter an integer N:"
read N

echo "The first $N prime numbers are:"

count=0
number=2

while [ $count -lt $N ]
do
  prime=1
  for (( i=2; i<$number; i++ ))
  do
    if [ $(($number % $i)) -eq 0 ]
    then
      prime=0
      break
    fi
  done

  if [ $prime -eq 1 ]
  then
    echo $number
    count=$(($count + 1))
  fi

  number=$(($number + 1))
done

```

Output

```
abir@abir-VirtualBox:~/Desktop/Shell_Script$ bash CG1.sh
Enter an integer N:
5
The first 5 prime numbers are:
2
3
5
7
11
```

2. Write a shell script to accept two files name and concatenate the content of the file into another file if both file are exits.

Answer:

Shell Script

```
#!/bin/bash

echo "Enter the name of the first file:"
read file1

echo "Enter the name of the second file:"
read file2

echo "Enter the name of the file to concatenate to:"
read concat_file

if [ -f $file1 ] && [ -f $file2 ]
then
  cat $file1 $file2 > $concat_file
  echo "Content of $file1 and $file2 have been concatenated to $concat_file."
else
  echo "Both files $file1 and $file2 must exist to concatenate their content."
fi

```

Output

```
abir@abir-VirtualBox:~/Desktop/Shell_Script$ sh CG2.sh
Enter the name of the first file:
file1
Enter the name of the second file:
file2
Enter the name of the file to concatenate to:
file3
Content of file1 and file2 have been concatenated to file3.
```

3. Write a shell script to list the attributes of the process running on a system, also store this information in a file named "processInfo.info".

Answer:

Shell Script

```
#!/bin/bash

echo "Listing process attributes..."

# get process ID
echo -n "Process ID: "
echo $$

# get process name
echo -n "Process name: "
echo $(basename $0)
```

```

# get process user
echo -n "Process user: "
echo $USER

# get process start time
echo -n "Process start time: "
echo $(date +"%Y-%m-%d %H:%M:%S")

# get process CPU usage
echo -n "Process CPU usage: "
echo $(ps -p $$ -o %cpu | awk 'NR==2')

# get process memory usage
echo -n "Process memory usage: "
echo $(ps -p $$ -o %mem | awk 'NR==2')

# write process information to file
echo "Writing process information to file processInfo.info..."
echo "Process ID: $$" > processInfo.info
echo "Process name: $(basename $0)" >> processInfo.info
echo "Process user: $USER" >> processInfo.info
echo "Process start time: $(date +"%Y-%m-%d %H:%M:%S")" >> processInfo.info
echo "Process CPU usage: $(ps -p $$ -o %cpu | awk 'NR==2')" >> processInfo.info
echo "Process memory usage: $(ps -p $$ -o %mem | awk 'NR==2')" >> processInfo.info

echo "Process attributes have been listed and written to processInfo.info."
```

Output

```
abir@abir-VirtualBox:~/Desktop/Shell_Script$ sh CG3.sh
Listing process attributes...
Process ID: 3187
Process name: CG3.sh
Process user: abir
Process start time: 2023-03-01 10:50:55
Process CPU usage: 0.0
Process memory usage: 0.0
Writing process information to file processInfo.info...
Process attributes have been listed and written to processInfo.info.
```

4. Write a shell script to check your present working directory, and make a list of a files modified within last 24 hours.

Answer:

Shell Script

```
#!/bin/bash

echo "Checking present working directory..."

# get the present working directory
pwd
```

```bash

echo "Listing files modified within the last 24 hours..."

# use the find command to list files modified within the last 24 hours
find . -type f -mtime -1 -ls > modified_files.txt

echo "List of modified files written to modified_files.txt."

```

Output

```
abir@abir-VirtualBox:~/Desktop/Shell_Script$ sh CG4.sh
Checking present working directory...
/home/abir/Desktop/Shell_Script
Listing files modified within the last 24 hours...
List of modified files written to modified_files.txt.
```

5. Write a shell script to check whether the number is prime or not. The number should be given as an input through command line argument.

Answer:

Shell Script

```bash
#!/bin/bash

# check if number of arguments is correct
if [ $# -ne 1 ]
then
  echo "Usage: $0 <number>"
  exit 1
fi

# get the input number
n=$1

# check if number is less than 2
if [ $n -lt 2 ]
then
  echo "$n is not a prime number"
  exit 0
fi

# check if number is equal to 2
if [ $n -eq 2 ]
then
  echo "$n is a prime number"
  exit 0
fi

# check if number is divisible by any integer from 2 to its square root
for ((i=2; i*i<=n; i++))
do
  if [ $((n%i)) -eq 0 ]
  then
```

```
    echo "$n is not a prime number"
    exit 0
  fi
done

# if none of the above conditions were met, the number is prime
echo "$n is a prime number"

```

Output

```
abir@abir-VirtualBox:~/Desktop/Shell_Script$ chmod u+x CG5.sh
abir@abir-VirtualBox:~/Desktop/Shell_Script$ ./CG5.sh 11
11 is a prime number
```

6. Write a shell script to count number of digit of a multi-digit number taken as command line input.

Answer:

Shell Script

```
#!/bin/bash

# check if number of arguments is correct
if [ $# -ne 1 ]
then
  echo "Usage: $0 <number>"
  exit 1
fi

# get the input number
n=$1

# check if number is negative
if [ $n -lt 0 ]
then
  n=$(( -n ))
fi

# count the number of digits in the number
count=0
while [ $n -gt 0 ]
do
  n=$(( n / 10 ))
  count=$(( count + 1 ))
done

# output the number of digits
echo "The number of digits in $1 is $count"

```

Output

```
abir@abir-VirtualBox:~/Desktop/Shell_Script$ chmod u+x CG6.sh
abir@abir-VirtualBox:~/Desktop/Shell_Script$ ./CG6.sh 22221
The number of digits in 22221 is 5
```

7. Write a shell script to create a substring from a string. The starting position and length of the substring must be given by the user.

Answer:

Shell Script

```bash
#!/bin/bash

# check if number of arguments is correct
if [ $# -ne 3 ]
then
  echo "Usage: $0 <string> <start position> <length>"
  exit 1
fi

# get the input string, start position, and length
string=$1
start_pos=$2
length=$3

# check if start position is negative
if [ $start_pos -lt 0 ]
then
  echo "Start position cannot be negative"
  exit 1
fi

# check if start position is greater than the length of the string
if [ $start_pos -gt ${#string} ]
then
  echo "Start position cannot be greater than the length of the string"
  exit 1
fi

# check if length is negative
if [ $length -lt 0 ]
then
  echo "Length cannot be negative"
  exit 1
fi

# create the substring
substring=${string:start_pos:length}

# output the substring
echo "The substring of \"$string\" starting from position $start_pos and of length $length is \"$substring\""

```

Output

```
abir@abir-VirtualBox:~/Desktop/Shell_Script$ chmod u+x CG7.sh
abir@abir-VirtualBox:~/Desktop/Shell_Script$ ./CG7.sh AbirBarman 2 4
The substring of "AbirBarman" starting from position 2 and of length 4
is "irBa"
```

8. Write a shell script to compare two files and list the dissimilarities.

Answer:

Shell Script

```bash
#!/bin/bash

# check if number of arguments is correct
if [ $# -ne 2 ]
then
  echo "Usage: $0 <file1> <file2>"
  exit 1
fi

# check if both files exist
if [ ! -f "$1" ] || [ ! -f "$2" ]
then
  echo "Error: Both files must exist"
  exit 1
fi

# compare the two files and output the differences
diff "$1" "$2"

```

Output

```
abir@abir-VirtualBox:~/Desktop/Shell_Script$ chmod u+x CG8.sh
abir@abir-VirtualBox:~/Desktop/Shell_Script$ ./CG8.sh file1 file2
1,12c1
< Country list
<
< Afghanistan
< Albania
< Algeria
< Andorra
< Angola
< Antigua and Barbuda
< Argentina
< Armenia
< Australia
< Austria
---
> capital List
13a3,12
> Kabul
> Tirana (Tirane)
> Algiers
> Andorra la Vella
> Luanda
> Saint John's
> Buenos Aires
> Yerevan
> Canberra
> Vienna
```

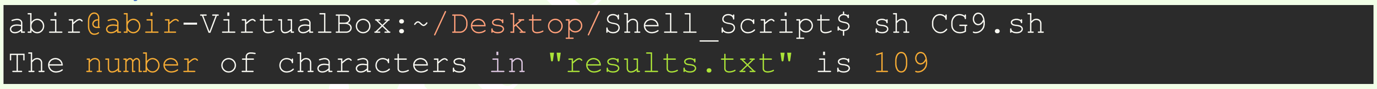
9. Write a shell script to get the current date, time, username and present working directory. Store the results in a file and count the number of characters in the file.

Answer:

Shell Script

```
#!/bin/bash

# get the current date and time
date_time=$(date "+%Y-%m-%d %H:%M:%S")

# get the username
username=$(whoami)

# get the present working directory
pwd=$(pwd)

# create a file and write the results to it
file="results.txt"
echo "Date and time: $date_time" >> "$file"
echo "Username: $username" >> "$file"
echo "Present working directory: $pwd" >> "$file"

# count the number of characters in the file
char_count=$(wc -m < "$file")

# output the number of characters in the file
echo "The number of characters in \"$file\" is $char_count"

```

Output



```
abir@abir-VirtualBox:~/Desktop/Shell_Script$ sh CG9.sh
The number of characters in "results.txt" is 109
```

10. Write a shell script to print the Fibonacci series up to N terms. Where N should be taken as command line input.

Answer:

Shell Script

```
#!/bin/bash

# check if number of arguments is correct
if [ $# -ne 1 ]
then
  echo "Usage: $0 <N>"
  exit 1
fi

# check if argument is a positive integer
if ! [[ "$1" =~ ^[1-9][0-9]*$ ]]
then
  echo "Error: Argument must be a positive integer"
  exit 1
fi

# initialize variables
n=$1
a=0
b=1

# print the first two terms of the series
echo -n "$a $b "

# print the remaining terms of the series
for (( i=3; i<=n; i++ ))
do
  # calculate the next term of the series
  c=$((a + b))
  echo -n "$c "

  # update the variables for the next iteration
  a=$b
  b=$c
done

echo # print a newline at the end
```

Output

```
abir@abir-VirtualBox:~/Desktop/Shell_Script$ chmod u+x CG10.sh
abir@abir-VirtualBox:~/Desktop/Shell_Script$ ./CG10.sh 10
0 1 1 2 3 5 8 13 21 34
```

11. Write a shell script to check whether a number is Armstrong number or not. The number should be given as an input through command line argument.

Answer:

Shell Script

```
#!/bin/bash

# check if number of arguments is correct
if [ $# -ne 1 ]
then
  echo "Usage: $0 <number>"
  exit 1
fi

# check if argument is a positive integer
if ! [[ "$1" =~ ^[1-9][0-9]*$ ]]
then
  echo "Error: Argument must be a positive integer"
  exit 1
fi
```

```bash

# initialize variables
number=$1
sum=0

# calculate the number of digits in the number
n=${#number}

# calculate the sum of the cubes of the digits
for (( i=0; i<$n; i++ ))
do
  digit=${number:$i:1}
  sum=$((sum + digit ** n))
done

# check if the number is an Armstrong number
if [ $sum -eq $number ]
then
  echo "$number is an Armstrong number"
else
  echo "$number is not an Armstrong number"
fi


```

Output

```
abir@abir-VirtualBox:~/Desktop/Shell_Script$ chmod u+x CG11.sh
abir@abir-VirtualBox:~/Desktop/Shell_Script$ ./CG11.sh 141
141 is not an Armstrong number
abir@abir-VirtualBox:~/Desktop/Shell_Script$ ./CG11.sh 1634
1634 is an Armstrong number
```

12. Write a shell script to sort an array of integers. The length of the array should be taken a command line argument and array elements should be provided at the run time.

Answer:

Shell Script

```bash
#!/bin/bash

# check if number of arguments is correct
if [ $# -ne 1 ]
then
  echo "Usage: $0 <length>"
  exit 1
fi

# check if argument is a positive integer
if ! [[ "$1" =~ ^[1-9][0-9]*$ ]]
then
  echo "Error: Argument must be a positive integer"
  exit 1
fi
```

```bash

# read array elements from user input
echo "Enter the array elements:"
for (( i=0; i<$1; i++ ))
do
  read -p "Element $((i+1)): " array[$i]
done

# sort the array using bubble sort algorithm
for (( i=0; i<$1-1; i++ ))
do
  for (( j=0; j<$1-i-1; j++ ))
  do
    if [ ${array[j]} -gt ${array[j+1]} ]
    then
      temp=${array[j]}
      array[j]=${array[j+1]}
      array[j+1]=$temp
    fi
  done
done

# print the sorted array
echo "Sorted array:"
for (( i=0; i<$1; i++ ))
do
  echo ${array[i]}
done


```

Output

```
abir@abir-VirtualBox:~/Desktop/Shell_Script$ chmod u+x CG12.sh
abir@abir-VirtualBox:~/Desktop/Shell_Script$ ./CG12.sh 5
Enter the array elements:
Element 1: 55
Element 2: 25
Element 3: 45
Element 4: 66
Element 5: 99
Sorted array:
25
45
55
66
99
```

13. Write a shell script to create an array using user input and perform scalar multiplication on the array elements.

Answer:

Shell Script

```bash
#!/bin/bash

# read array size from user input
read -p "Enter the size of the array: " size

# declare an array of the given size
declare -a array

# read array elements from user input
echo "Enter the array elements:"
for (( i=0; i<$size; i++ ))
do
  read -p "Element $((i+1)): " array[$i]
done

# read scalar value from user input
read -p "Enter the scalar value: " scalar

# perform scalar multiplication on the array elements
for (( i=0; i<$size; i++ ))
do
  array[$i]=$((array[$i]*scalar))
done

# print the resulting array
echo "Resulting array:"
for (( i=0; i<$size; i++ ))
do
  echo ${array[$i]}
done

```

Output

```
abir@abir-VirtualBox:~/Desktop/Shell_Script$ bash CG13.sh
Enter the size of the array: 3
Enter the array elements:
Element 1: 5
Element 2: 6
Element 3: 7
Enter the scalar value: 2
Resulting array:
10
12
14
```

14. Write a shell script to input a file name containing multiple lines and then create a new file by copying line 4 to 10 from inputted file.

Answer:

Shell Script

```bash
#!/bin/bash

# read input file name from user input
read -p "Enter the input file name: " input_file

# check if input file exists
if [ ! -f $input_file ]
then
  echo "Error: Input file $input_file does not exist"
  exit 1
fi

# create output file name
output_file="${input_file}_output"

# copy lines 4 to 10 from input file to output file
sed -n '4,10p' $input_file > $output_file

# print success message
echo "Lines 4 to 10 from $input_file have been copied to $output_file"

```

Output

```
abir@abir-VirtualBox:~/Desktop/Shell_Script$ sh CG14.sh
Enter the input file name: file1
Lines 4 to 10 from file1 have been copied to file1_output
```

15. Write a shell script to input a year as command line argument and check whether the year is palindrome or not.

Answer:

Shell Script

```bash
#!/bin/bash

# check if year argument is passed
if [ $# -ne 1 ]
then
  echo "Error: Please provide a year as command line argument"
  exit 1
fi

# store year in a variable
year=$1

# reverse the year
reverse=0
temp=$year
```

```
while [ $temp -gt 0 ]
do
  remainder=$((temp % 10))
  reverse=$((reverse * 10 + remainder))
  temp=$((temp / 10))
done

# check if year is a palindrome
if [ $year -eq $reverse ]
then
  echo "$year is a palindrome"
else
  echo "$year is not a palindrome"
fi


```

Output

```
abir@abir-VirtualBox:~/Desktop/Shell_Script$ chmod u+x CG15.sh
abir@abir-VirtualBox:~/Desktop/Shell_Script$ ./CG15.sh 141
141 is a palindrome
```

16. Write a shell script to find whether a pair of integers are twin prime. A twin prime is either a prime number that is 2 less or 2 more than another prime number- for example, member of the twin prime pair (41, 43). In other words, a twin prime is a prime that has a prime gap of two.

Answer:

Shell Script

```
#!/bin/bash

# Check if two arguments are passed
if [ $# -ne 2 ]
then
  echo "Error: Please provide two integers as command line arguments."
  exit 1
fi

# Store the two arguments as variables
num1=$1
num2=$2

# Check if the two numbers are prime
is_prime()
{
  n=$1
  if [ $n -lt 2 ]
  then
    return 1
  fi
  for (( i=2; i<=n/2; i++ ))
  do
    if [ $((n%i)) -eq 0 ]
    then
      return 1
    fi
  done
  return 0
}

is_prime $num1
if [ $? -ne 0 ]
then
  echo "$num1 is not a prime number."
  exit 1
fi

is_prime $num2
if [ $? -ne 0 ]
then
  echo "$num2 is not a prime number."
  exit 1
fi

# Check if the two numbers are twin primes
diff=$((num2-num1))
if [ $diff -eq 2 ] || [ $diff -eq -2 ]
then
  echo "$num1 and $num2 are twin primes."
else
  echo "$num1 and $num2 are not twin primes."
fi


```

Output

```
abir@abir-VirtualBox:~/Desktop/Shell_Script$ chmod u+x CG16.sh
abir@abir-VirtualBox:~/Desktop/Shell_Script$ ./CG16.sh 41 43
41 and 43 are twin primes.
```

17. Write a shell script to count number of words in a string given by the user.

Answer:

Shell Script

```
#!/bin/bash

echo "Enter a string:"
read input_string

word_count=0

# loop through the string and count the number of spaces
```

```
for (( i=0; i<${#input_string}; i++ )); do
    if [[ "${input_string:i:1}" == " " ]]; then
        (( word_count++ ))
    fi
done

# add 1 to word count for the last word
(( word_count++ ))

echo "Number of words in the string: $word_count"

```

Output

```
abir@abir-VirtualBox:~/Desktop/Shell_Script$ bash CG17.sh
Enter a string:
A quick brown fox jumps over the lazy dog
Number of words in the string: 9
```

18. Write a shell script to check whether a number is BUZZ number or not. A BUZZ number is the number, which either ends with 7 or is divided by 7.

Answer:

Shell Script

```
#!/bin/bash

echo "Enter a number:"
read num

if (( num % 7 == 0 )) || (( num % 10 == 7 )); then
    echo "$num is a BUZZ number"
else
    echo "$num is not a BUZZ number"
fi

```

Output

```
abir@abir-VirtualBox:~/Desktop/Shell_Script$ bash CG18.sh
Enter a number:
70
70 is a BUZZ number
```

19. Write a shell script to find the GCD of two integers taken as command line arguments.

Answer:

Shell Script

```bash
#!/bin/bash

# Check if both arguments are provided
if [ $# -ne 2 ]; then
    echo "Usage: $0 num1 num2"
    exit 1
fi

# Get the arguments
num1=$1
num2=$2

# Find the GCD using Euclid's algorithm
while [ $num2 -ne 0 ]
do
    remainder=$(( num1 % num2 ))
    num1=$num2
    num2=$remainder
done

# Print the GCD
echo "GCD of $1 and $2 is: $num1"

```

Output

```
abir@abir-VirtualBox:~/Desktop/Shell_Script$ chmod u+x CG19.sh
abir@abir-VirtualBox:~/Desktop/Shell_Script$ ./CG19.sh 27 15
GCD of 27 and 15 is: 3
```

20. Write a shell script to find the LCM of two integers taken as command line arguments.

Answer:

Shell Script

```bash
#!/bin/bash

# Check if both arguments are provided
if [ $# -ne 2 ]; then
    echo "Usage: $0 num1 num2"
    exit 1
fi

# Get the arguments
num1=$1
num2=$2

# Find the GCD using Euclid's algorithm
```

```
while [ $num2 -ne 0 ]
do
    remainder=$(( num1 % num2 ))
    num1=$num2
    num2=$remainder
done

# Calculate the LCM
lcm=$(( $1 * $2 / $num1 ))

# Print the LCM
echo "LCM of $1 and $2 is: $lcm"

```

Output

```
abir@abir-VirtualBox:~/Desktop/Shell_Script$ ./CG20.sh 6 7
LCM of 6 and 7 is: 42
```

21. Write a shell script to check whether a number is automorphic number or not. An automorphic number contains the last digits of its square. 25 is an automorphic number as its square is 625 and 25 is present as the last two digit.

Answer:

Shell Script

```
#!/bin/bash

# Take the number as input from the user
echo "Enter a number: "
read num

# Compute the square of the number
square=$(expr $num \* $num)

# Extract the last digits of the square and the number
num_digits=${#num}
last_digits=${square:(-$num_digits)}

# Check if the last digits of the square match the original number
if [ "$last_digits" == "$num" ]; then
    echo "$num is an automorphic number"
else
    echo "$num is not an automorphic number"
fi

```

Output

```
abir@abir-VirtualBox:~/Desktop/Shell_Script$ bash CG21.sh
Enter a number:
25
25 is an automorphic number
```

22. Write a shell script to find the sum of digits from a multi-digit number.

Answer:

Shell Script

```bash
#!/bin/bash

num=$1    # get number from command line argument
sum=0    # initialize sum to zero

while [ $num -gt 0 ]; do
    digit=$((num % 10))   # get the last digit
    sum=$((sum + digit))  # add the digit to the sum
    num=$((num / 10))     # remove the last digit
done

echo "The sum of digits in $1 is: $sum"

```

Output

```
abir@abir-VirtualBox:~/Desktop/Shell_Script$ chmod u+x CG22.sh
abir@abir-VirtualBox:~/Desktop/Shell_Script$ ./CG22.sh 142
The sum of digits in 142 is: 7
```

23. Write a shell script to input a multi-digit number and print whether the number is special number or not. A number is said to be a special number if the sum of the factorial of the digits of the number is same as the original number. 145 is a special number as 1! + 4! + 5! =1 + 24 + 120 = 145.

Answer:

Shell Script

```bash
#!/bin/bash

echo "Enter a multi-digit number:"
read n

temp=$n
sum=0

while [ $temp -gt 0 ]
do
    digit=$((temp%10))
    fact=1
    for (( i=1; i<=digit; i++ ))
    do
        fact=$((fact*i))
    done
    sum=$((sum+fact))
    temp=$((temp/10))
done
```

```

if [ $n -eq $sum ]
then
    echo "$n is a special number."
else
    echo "$n is not a special number."
fi

```

Output

```
abir@abir-VirtualBox:~/Desktop/Shell_Script$ bash CG23.sh
Enter a multi-digit number:
145
145 is a special number.
```

24. Write a shell script to check whether a number is duck number or not. A number is said to be duck if the digit 0(zero) is present in it.

Answer:

Shell Script

```
#!/bin/bash

read -p "Enter a number: " number

if [[ "${number:0:1}" != "0" ]] && [[ "${number}" =~ "0" ]]; then
  echo "${number} is a duck number"
else
  echo "${number} is not a duck number"
fi

```

Output

```
abir@abir-VirtualBox:~/Desktop/Shell_Script$ bash CG24.sh
Enter a number: 202
202 is a duck number
```