**EXPERIMENT 2 Date: 07/08/23**

**Title: SHELL SCRIPT PROGRAMS**

**AIM:** To write Shell Script programs for the following:

1) To find maximum and minimum digit of a number.

2) To calculate square root of a number.

3) To convert Celsius to Fahrenheit.

4) To calculate simple interest.

5) To generate the following pattern

\*

\* \*

\* \* \*

\* \* \* \*

**THEORY:**

1] IF-ELSE STATEMENT:

* The if...else...fi statement is the next form of control statement that allows
* Shell to execute statements in a controlled way and make the right choice.
* The Shell expression is evaluated in the above syntax.
* If the resulting value is true, given statement(s) are executed.
* If the expression is false, then no statement will be executed.
* **SYNTAX:**

if [ expression ] then

Statement(s) to be executed if expression is true else

Statement(s) to be executed if expression is not true

Fi

* **Sample Program:**

Display number if greater than 5 else print less than five

echo “Enter the number”

read a

if [ $num -gt 5 ]

then

echo “$a is greater than 5”

else

echo” $a less than 5”

fi

2] WHILE STATEMENT:

* The while loop enables you to execute a set of commands repeatedly until some
* condition occurs.
* It is usually used when you need to manipulate the value of a variable
* repeatedly.
* Here the Shell command is evaluated.
* If the resulting value is true, given statement(s) are executed.
* If command is false then no statement will be executed and the program will jump
* to the next line after the done statement.
* **SYNTAX:**

while command

do

Statement(s) to be executed if command is true

Done

* **Sample program:**

Find the factorial of a number.

echo "Enter the number"

read a

fact=1

x=1

while [ $x -le $a ]

do

fact=$(($fact \* $x))

x=$(($x + 1))

done

echo "factorial is - $fact"

3] CASE CONDITION:

* Shell supports case...esac statement which handles exactly this situation,
* and it does so more efficiently than repeated if...elif statements.
* The basic syntax of the case...esac statement is to give an expression to
* evaluate and to execute several different statements based on the value of the
* expression.
* The interpreter checks each case against the value of the expression
* until a match is found.
* If nothing matches, a default condition will be used.
* Here the string word is compared against every pattern until a match is
* found.
* The statement(s) following the matching pattern executes.
* If no matches are found, the case statement exits without performing
* any action.
* There is no maximum number of patterns, but the minimum is one.
* When statement(s) part executes, the command (;;) indicates that the program
* flow should jump to the end of the entire case statement. This is similar to break
* in the C programming language.
* **SYNTAX:**

case word in

pattern1)

Statement(s) to be executed if pattern1 matches

;;

pattern2)

Statement(s) to be executed if pattern2 matches

;;

pattern3)

Statement(s) to be executed if pattern3 matches

;;

Default condition to be executed

;;

esac

* **Sample Program:**

Write a menu driven program to execute the given commands.

ch=1

while [ $ch -eq 1 ]

do

echo ”Menu\n”

echo”1. Todays date 2. Current working directory 3. Display the users of system”

echo ”Enter your choice”

read choice

case $choice in

1. date;;
2. pwd ;;
3. who;;

echo”Invalid choice:

esac

echo”do you want to continue”

Read ch

done

4] FOR LOOP:

* The for loop operate on lists of items.
* It repeats a set of commands for every item in a list.
* Here var is the name of a variable and word 1 to word N are sequences
* of characters separated by spaces (words).
* Each time the for loop executes, the value of the variable var is set to the next
* word in the list of words, word 1 to word N.
* **SYNTAX:**

for var in word 1 word 2 ...word n

do

Statement to be executed done

* **Sample Program:**

for ((c=1 ; c<=5; c++ ))

do

Echo”hi $c time”

**Programs:**

1. To find maximum and minimum digit of a number.

Code:

echo "Enter the number"

read n

max=-1

min=10

while [ $n -gt 0 ]

do

rem=$(($n%10))

if [ $rem -lt $min ]

then

min=$rem

fi

if [ $rem -gt $max ]

then

max=$rem

fi

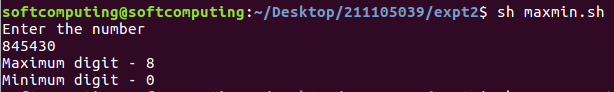
n=$(($n/10))

done

echo "Maximum digit - $max"

echo "Minimum digit - $min"

Output:



1. To calculate square root of a number.

Code:

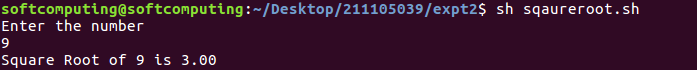
echo "Enter the number"

read n

root=$(echo "scale=2; sqrt($n)" | bc)

echo "Square Root of $n is $root"

Output:



1. To convert Celsius to Fahrenheit.

Code:

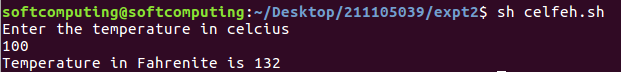
echo "Enter the temperature in celcius"

read c

f=$((($c\*(9/5)) + 32))

echo "Temperature in Fahrenite is $f"

Output:



1. To calculate simple interest.

Code:

echo "Enter the principal amount"

read p

echo "Enter the Rate"

read r

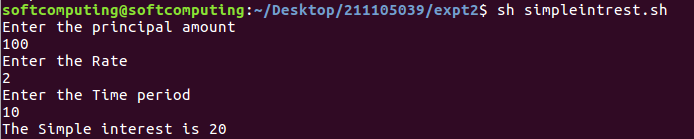
echo "Enter the Time period"

read t

si=$((($p\*$r\*$t)/100))

echo "The Simple interest is $si"

Output:



1. To generate the following pattern

Code:

echo "Enter the number of rows"

read n

m=$n

for ((i=1; i<=$n; i++))

do

for ((j=1; j <= $m - 1; j++))

do

printf " "

done

for ((k=1; k <= i; k++))

do

printf "\* "

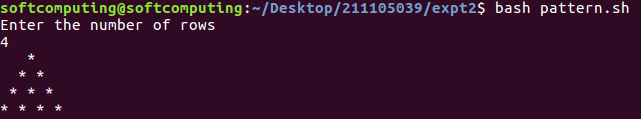
done

m=$(($m-1))

printf "\n"

done

Output:



**Conclusion:**

The Shell Script Programs were run and executed successfully.