**SHELL PROGRAMMING**

**Experiment No: 2 Date:07/08/23**

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

1) Generate square of a number.

2) Multiplication table of 9.

3) Menu driven program to calculate area of circle, triangle and rectangle.

4) Display numbers from 10-60 divisible by 4.

5) Second largest and second smallest digit of a number.

**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

* **Example 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

* **Example 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

* **Example 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

* **Example Program:**

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

do

Echo”hi $c time”

**Programs:**

1. Generate square of a number.

**Code:**

#!/bin/bash

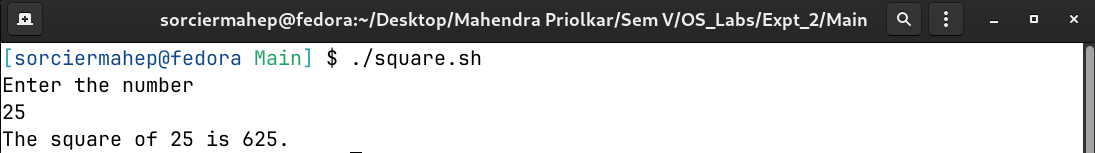
echo "Enter the number"

read num

square=$(($num \*\* 2))

echo "The square of $num is $square."

**Output:**



1. Multiplication table of 9.

**Code:**

#!/bin/bash

echo "Enter the number of multiples of 9"

read mults

mult=9

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

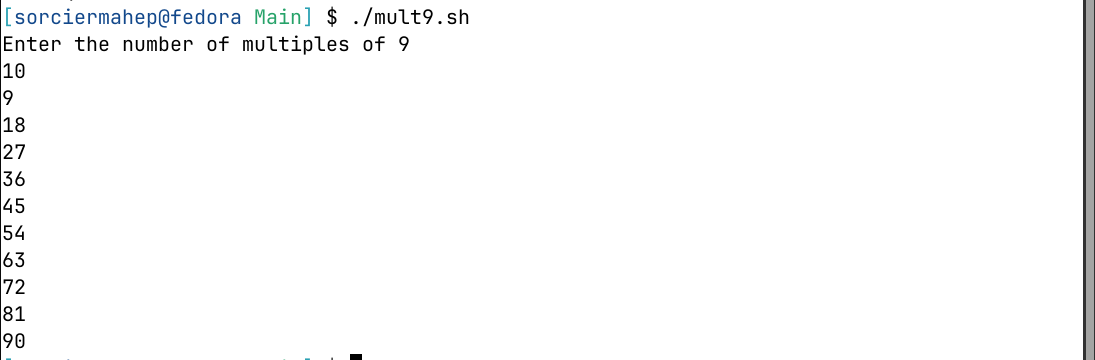
do

a=$(( $i \* $mult ))

printf "%d\n" "$a"

done

**Output:**



1. Menu driven program to calculate area of circle, triangle and rectangle.

**Code:**

#!/bin/bash

echo -e "Enter 1 for circle.\nEnter 2 for triangle.\nEnter 3 for rectangle.\n"

read choice

case $choice in

1)

echo "Enter radius of circle"

read r

PI=3.14159

ans=$(echo "scale=3;$PI\*$r\*$r"|bc )

echo "$ans"

;;

2)

echo "Enter base and height of triangle"

read b h

ans=$(($b\*$h/2))

echo "$ans"

;;

3)

echo "Enter sides of rectangle"

read b l

ans=$(($b\*$l))

echo "$ans"

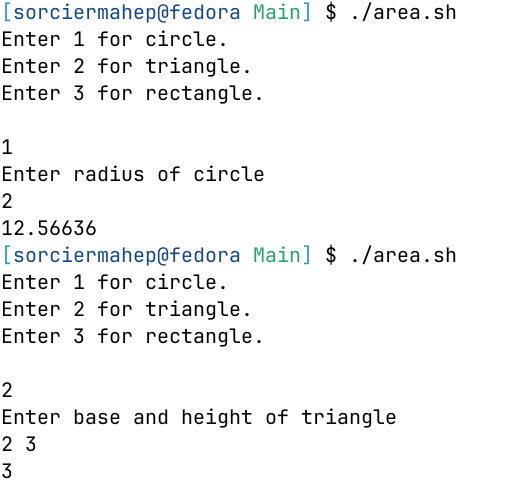
;;

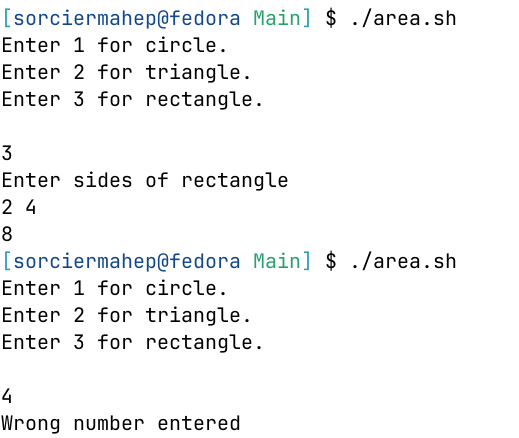
\*)

echo "Wrong number entered"

esac

**Output:**





1. Display numbers from 10-60 divisible by 4.

**Code:**

#!/bin/bash

for((i=10;i<=60;i++))

do

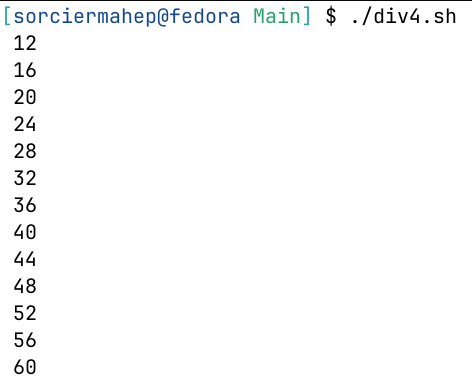
if !(($i%4)); then

echo " $i "

fi

done

**Output:**



1. Second largest and second smallest digit of a number.

**Code:**

#!/bin/bash

echo "Enter the number"

read num

sec\_small=10

small=10

sec\_large=-1

large=-1

for((i=0;i<${#num};i++))

do

if [ ${num:i:1} -gt $large ]; then

sec\_large=$large

large=${num:i:1}

fi

if [ ${num:i:1} -lt $large ] && [ ${num:i:1} -gt $sec\_large ]; then

sec\_large=${num:i:1}

fi

if [ ${num:i:1} -lt $small ]; then

sec\_small=$small

small=${num:i:1}

fi

if [ ${num:i:1} -gt $small ] && [ ${num:i:1} -lt $sec\_small ]; then

sec\_small=${num:i:1}

fi

done

if [ $sec\_large -eq -1 ] || [ $sec\_small -eq 10 ]; then

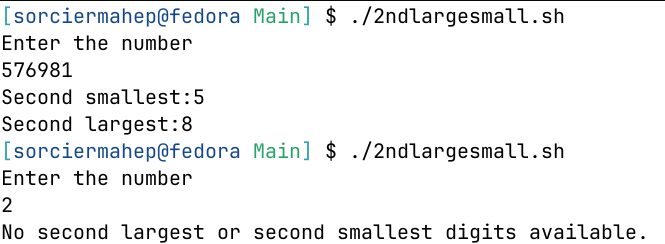
echo "No second largest or second smallest digits available."

else

echo -e "Second smallest:$sec\_small\nSecond largest:$sec\_large"

fi

**Output:**



**Conclusion:**

The Shell Script Programs were run and executed successfully.