Experiment 4

Write and implement a shell script to perform basic arithmetic operations (addition, subtraction, multiplication, division) based on user input.

Fundamentals

Problem: Define two variables, then display the values of variables

```
var1=10
var2=20
echo $var1 $var2
```

Note: We can define a variable by using the syntax ***variable_name = value***. To get the value of the variable, add ***$*** before the variable.

Problem: Ask for name of person and print "Hello, person_name" message to that person

```
echo "What is your name?"
read PERSON
echo "Hello, $PERSON"
```

```
read -p "Enter your name" PERSON
echo "Hello, $PERSON"
```

Problem: Take two variables name and surname and print full name.

```c
#include <stdio.h>

int main() {

    char name[50];
    char surname[50];

    printf("Enter your name: ");
    scanf("%s", name);
    printf("Enter your surname: ");
    scanf("%s", surname);

    printf("Full name: %s %s\n", name, surname);

    return 0;
}
```

C Program

```bash
#!/bin/bash

echo -n "Enter your first name: "
read name

echo -n "Enter your surname: "
read surname

echo "Full name: $name $surname"
```

Shell Program

Assume variable a holds 10 and variable b holds 20 then −

Operator	**Description**	**Example**
+ (Addition)	Adds values on either side of the operator	`expr $a + $b` will give 30
- (Subtraction)	Subtracts right hand operand from left hand operand	`expr $a - $b` will give -10
* (Multiplication)	Multiplies values on either side of the operator	`expr $a \* $b` will give 200
/ (Division)	Divides left hand operand by right hand operand	`expr $b / $a` will give 2
% (Modulus)	Divides left hand operand by right hand operand and returns remainder	`expr $b % $a` will give 0

Arithmetic Operators

Operator	Description	Example
-eq	Checks if the value of two operands are equal or not; if yes, then the condition becomes true.	[$a -eq $b]
-ne	Checks if the value of two operands are equal or not; if values are not equal, then the condition becomes true.	[$a -ne $b]
-gt	Checks if the value of left operand is greater than the value of right operand; if yes, then the condition becomes true.	[$a -gt $b]
-lt	Checks if the value of left operand is less than the value of right operand; if yes, then the condition becomes true.	[$a -lt $b]
-ge	Checks if the value of left operand is greater than or equal to the value of right operand; if yes, then the condition becomes true.	[$a -ge $b]
-le	Checks if the value of left operand is less than or equal to the value of right operand; if yes, then the condition becomes true.	[$a -le $b]

Logical Operators

Conditional Statements

Format

```
if [ expression ]
then
        Statement(s) to be executed if expression is true
fi
```

Example

```
a=10
b=20

if [ $a -eq $b ]
then
    echo "a is equal to b"
fi
```

Format

```
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

```
a=10
b=20

if [ $a -eq $b ]
then
     echo "a is equal to b"
else
     echo "a is not equal to b"
fi
```

Format

```
if [expression 1]
then
     Statement(s) to be executed if expression 1 is true
elif [expression 2]
then
     Statement(s) to be executed if expression 2 is true
elif [expression 3]
then
     Statement(s) to be executed if expression 3 is true
Else
     Statement(s) to be executed if no expression is true
fi
```

Example

```
a=10
b=20

if [ $a -eq $b ]
then
     echo "a is equal to b"
elif [ $a -gt $b ]
then
     echo "a is greater than b"
elif [ $a -lt $b ]
then
     echo "a is less than b"
else
     echo "None of the condition met"
fi
```

Format

```
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

```
FRUIT="kiwi"

case "$FRUIT" in
     "apple")
           echo "Apple pie is quite tasty."
           ;;
     "banana")
           echo "I like banana nut bread."
           ;;
     "kiwi")
           echo "New Zealand is famous for kiwi."
           ;;
esac
```

Loop Statements

Format

```
while [condition]
do
     Statement(s) to be executed if command is true
done
```

Example

```
a=0

while [ $a -lt 10 ]
do
   echo $a
   a=`expr $a + 1`
done
```

Format

```
for var in word1 word2 ... wordN
do
     Statement(s) to be executed for every word.
done
```

Example

```
for var in 0 1 2 3 4 5 6 7 8 9
do
     echo $var
done
```

Format

```
until [condition]
do
     Statement(s) to be executed until command is true
done
```

Example

```
a=0

until [ ! $a -lt 10 ]
do
   echo $a
   a=`expr $a + 1`
done
```

Perform basic arithmetic operations based on user input.

Algorithm

```
1. Start
2. Input Operation Type:
    • Display a prompt: “Enter the arithmetic operation(+, -, *, /):”
    • Read the input as operation.
3. Input Numbers:
    • Display a prompt: “Enter the first number:”
    • Read the input as num1.
    • Display a prompt: “Enter the second number:”
    • Read the input as num2.
4. Evaluate the Operation:
    • If operation is "+":
        • Calculate result = num1 + num2.
        • Display: “The sum is: result”
    • Else if operation is "-":
        • Calculate result = num1 - num2.
        • Display: "The difference is: result".
    • Else if operation is "*":
        • Calculate result = num1 * num2.
        • Display: "The product is: result".
    • Else if operation is "/":
        • If num2 is 0:
            • Display: "Error: Division by zero is not allowed."
        • Else:
            • Calculate result = num1 / num2.
            • Display: "The quotient is: result".
    • Else:
        • Display: “Invalid operation. Please enter one of +, -, *, /.”
5. End
```

C Program

```c
#include <stdio.h>

int main()
{
    char operation;
    double num1, num2, result;

    // Prompt the user for the operation type
    printf("Enter the arithmetic operation (+, -, *, /): ");
    scanf("%c", &operation);

    // Prompt the user for the two numbers
    printf("Enter the first number: ");
    scanf("%lf", &num1);

    printf("Enter the second number: ");
    scanf("%lf", &num2);
```

```
// Perform the selected operation
    if (operation == '+’)
    {
        result = num1 + num2;
        printf("The sum is: %.2lf\n", result);
    }
    else if (operation == '-’)
    {
        result = num1 - num2;
        printf("The difference is: %.2lf\n", result);
    }
    else if (operation == '*’)
    {
        result = num1 * num2;
        printf("The product is: %.2lf\n", result);
    }
```

```c
    else if (operation == '/’)
    {
        if (num2 == 0)
        {
            printf("Error: Division by zero is not allowed.\n");
        }
        else
        {
            result = num1 / num2;
            printf("The quotient is: %.2lf\n", result);
        }
    }
    else
    {
        printf("Invalid operation. Please enter one of +, -, *, /.\n");
    }

    return 0;
}
```

Shell Program

```
#!/bin/bash

# Prompt the user for the operation type
echo "Enter the arithmetic operation
       (+, -, *, /):"
read operation

# Prompt the user for the two numbers
echo "Enter the first number:"
read num1

echo "Enter the second number:"
read num2

# Perform the selected operation
if [ "$operation" = "+" ]; then
    result=$((num1 + num2))
    echo "The sum is: $result"

elif [ "$operation" = "-" ]; then
    result=$((num1 - num2))
    echo "The difference is: $result"
```

```
elif [ "$operation" = "*" ]; then
    result=$((num1 * num2))
    echo "The product is: $result"

elif [ "$operation" = "/" ]; then
    if [ "$num2" -eq 0 ]; then
        echo "Error: Division by zero is
               not allowed."
    else
        result=$((num1 / num2))
        echo "The quotient is: $result"
    fi

else
    echo "Invalid operation. Please enter
           one of +, -, *, /."
fi
```