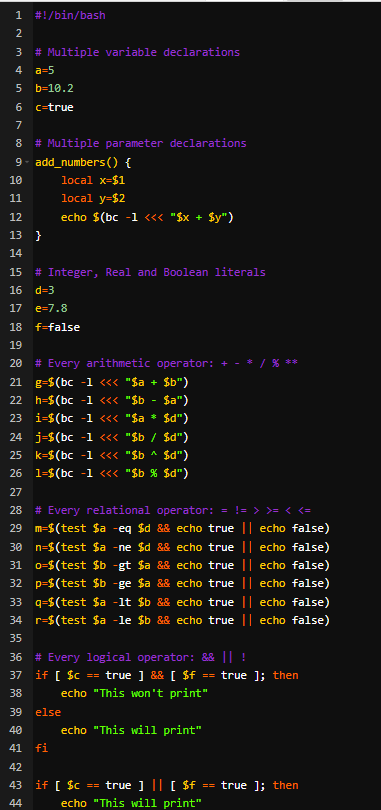
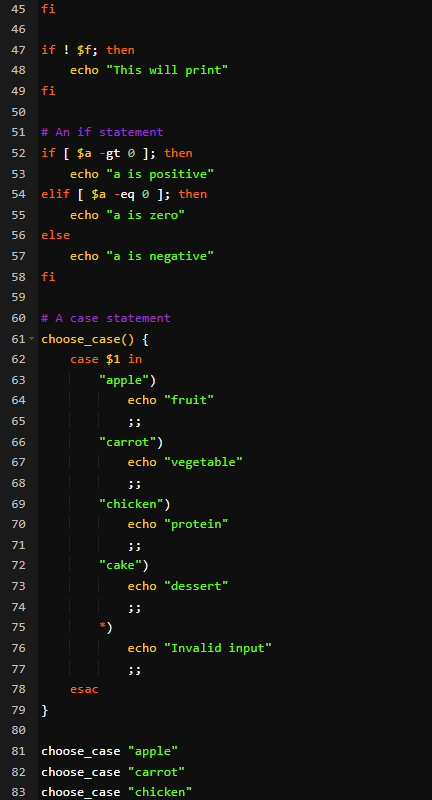


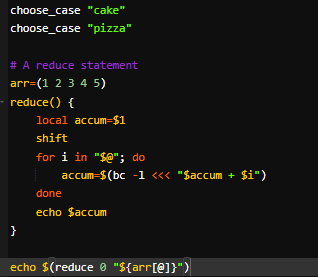
**Step 1: Algorithm**

1. Declare three variables: a as an integer with value 5, b as a real number with value 10.2, and c as a boolean with value true.
2. Define a function add\_numbers that takes two parameters, x and y. Inside the function, assign the values of x and y to local variables local x=$1 and local y=$2. Then, use the bc command to perform the addition of x and y and return the result using echo.
3. Declare three variables d as an integer with value 3, e as a real number with value 7.8, and f as a boolean with value false.
4. Use bc command to perform arithmetic operations on a, b, d, and store the results in the variables g, h, i, j, k, and l, respectively.
5. Use the test command and comparison operators to perform relational operations on a and d, and store the results in the variable m. Similarly, store the results of other relational operations on b and a in o and p, respectively. Store the results of the relational operation on a and b in q, and store the results of the relational operation on a and b in r.
6. Use the if statement with logical operators to perform logical operations on c and f, and print messages based on the results of the logical operations.
7. Use the if-elif-else statement to print messages based on the value of a.
8. Define a function choose\_case that takes a parameter input. Use the case statement to match input to one of the predefined cases and print the corresponding message. Print an error message if input does not match any of the cases.
9. Define a function reduce that takes two parameters, accum and arr. Inside the function, assign the value of accum to a local variable accum=$1. Use a for loop to iterate through the elements of arr. Use the bc command to perform addition operations on accum and the current element of arr. Return the final value of accum.
10. Call the choose\_case function with different inputs to test its functionality.
11. Call the reduce function with an array and print its result.
12. **Step 2: Code**
13. #!/bin/bash
14. # Multiple variable declarations  
    a=5  
    b=10.2  
    c=true
15. # Multiple parameter declarations  
    add\_numbers() {  
        local x=$1  
        local y=$2  
        echo $(bc -l <<< "$x + $y")  
    }
16. # Integer, Real and Boolean literals  
    d=3  
    e=7.8  
    f=false
17. # Every arithmetic operator: + - \* / % \*\*  
    g=$(bc -l <<< "$a + $b")  
    h=$(bc -l <<< "$b - $a")  
    i=$(bc -l <<< "$a \* $d")  
    j=$(bc -l <<< "$b / $d")  
    k=$(bc -l <<< "$b ^ $d")  
    l=$(bc -l <<< "$b % $d")
18. # Every relational operator: = != > >= < <=  
    m=$(test $a -eq $d && echo true || echo false)  
    n=$(test $a -ne $d && echo true || echo false)  
    o=$(test $b -gt $a && echo true || echo false)  
    p=$(test $b -ge $a && echo true || echo false)  
    q=$(test $a -lt $b && echo true || echo false)  
    r=$(test $a -le $b && echo true || echo false)
19. # Every logical operator: && || !  
    if [ $c == true ] && [ $f == true ]; then  
        echo "This won't print"  
    else  
        echo "This will print"  
    fi
20. if [ $c == true ] || [ $f == true ]; then  
        echo "This will print"  
    fi
21. if ! $f; then  
        echo "This will print"  
    fi
22. # An if statement  
    if [ $a -gt 0 ]; then  
        echo "a is positive"  
    elif [ $a -eq 0 ]; then  
        echo "a is zero"  
    else  
        echo "a is negative"  
    fi
23. # A case statement  
    choose\_case() {  
        case $1 in  
            "apple")  
                echo "fruit"  
                ;;  
            "carrot")  
                echo "vegetable"  
                ;;  
            "chicken")  
                echo "protein"  
                ;;  
            "cake")  
                echo "dessert"  
                ;;  
            \*)  
                echo "Invalid input"  
                ;;  
        esac  
    }
24. choose\_case "apple"  
    choose\_case "carrot"  
    choose\_case "chicken"  
    choose\_case "cake"  
    choose\_case "pizza"
25. # A reduce statement  
    arr=(1 2 3 4 5)  
    reduce() {  
        local accum=$1  
        shift  
        for i in "$@"; do  
            accum=$(bc -l <<< "$accum + $i")  
        done  
        echo $accum  
    }
26. echo $(reduce 0 "${arr[@]}")

**Step 3: Code screenshot:**

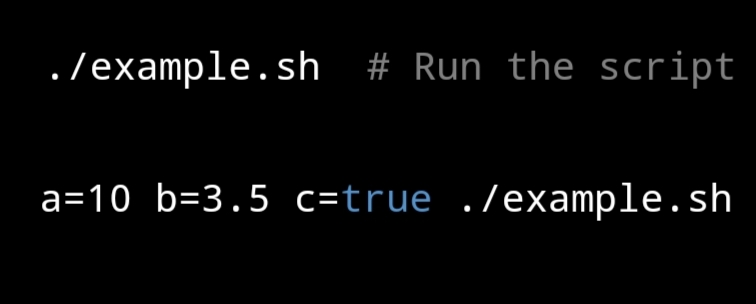






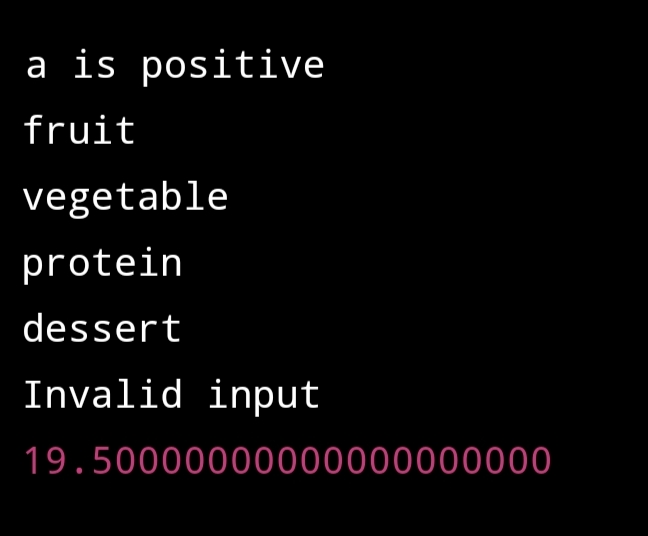
**Solution**

#To test the associativity and precedence of the operators, you can use different combinations of variables and literals with the arithmetic, relational, and logical operators, and observe the results. You can also use the echo command to print out the intermediate results. Here are some sample inputs:



**Output**:

The output of the program when it is run with the input values a=10 b=3.5 c=true:



**References:**

Bash Reference Manual: https://www.gnu.org/savannah-checkouts/gnu/bash/manual/bash.html

Bash Guide for Beginners: https://tldp.org/LDP/Bash-Beginners-Guide/html/

Advanced Bash-Scripting Guide: http://www.tldp.org/LDP/abs/html/

Awk User's Guide: https://www.gnu.org/software/gawk/manual/gawk.html