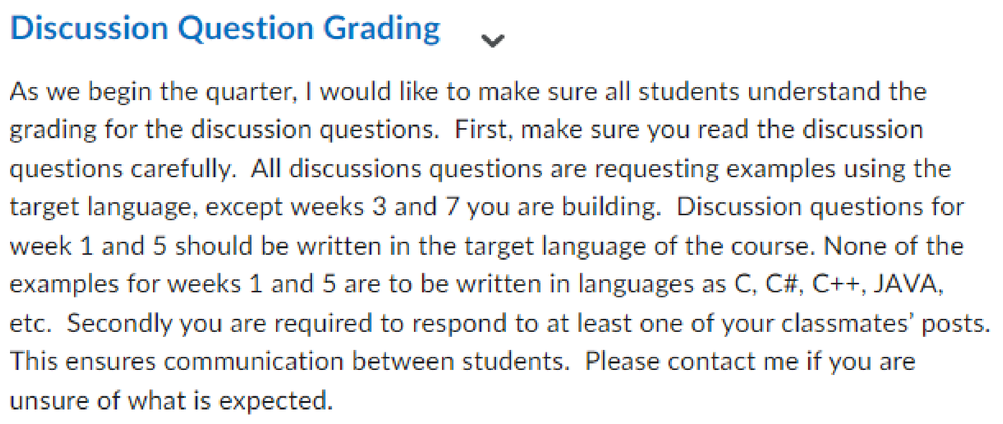
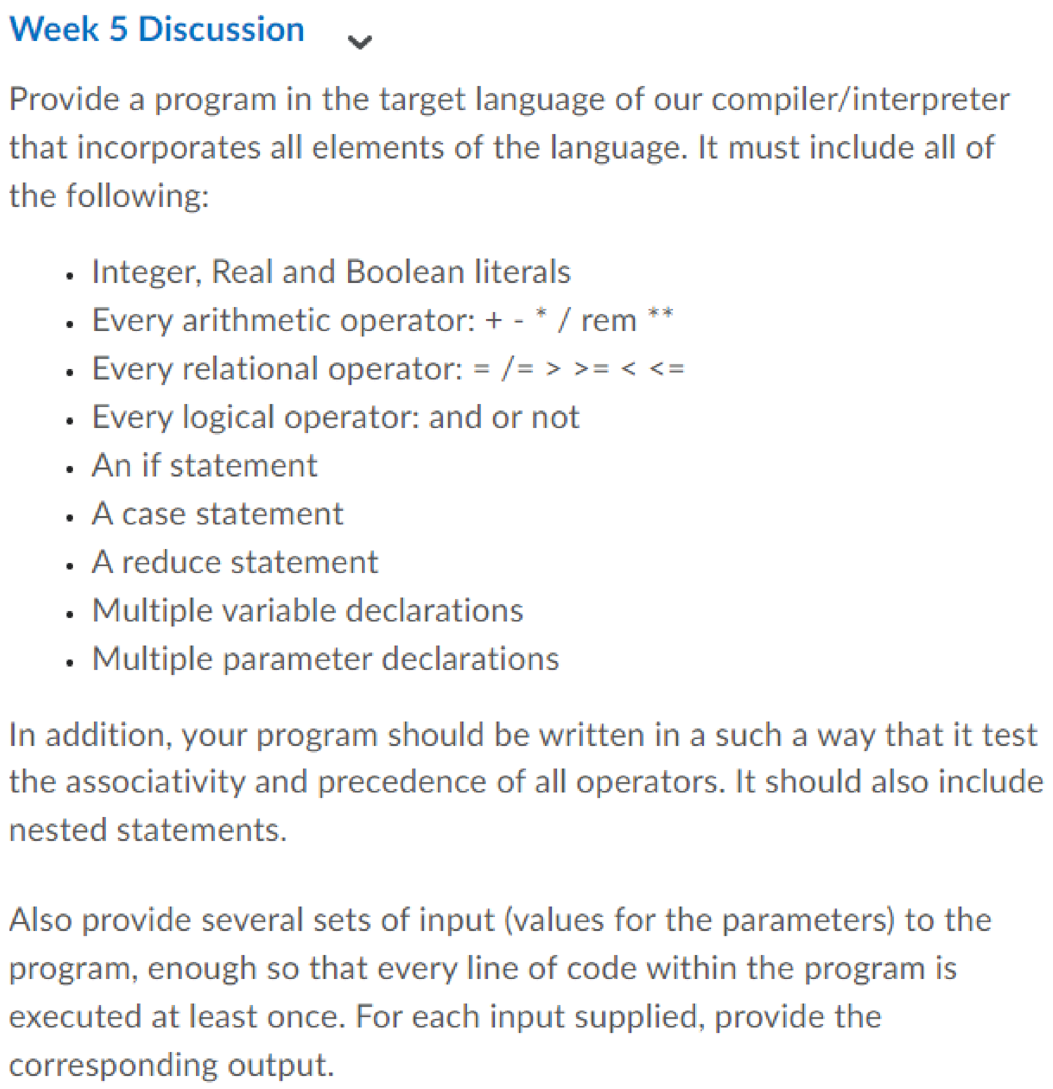
# Discussion Question Week 5



## Discussion Question Grading

As we begin the quarter, I would like to make sure all students understand the grading for the discussion questions. First, make sure you read the discussion questions carefully. All discussions questions are requesting examples using the target language, except weeks 3 and 7 you are building. Discussion questions for week 1 and 5 should be written in the target language of the course. None of the examples for weeks 1 and 5 are to be written in languages as C, C#, C++, JAVA, etc. Secondly you are required to respond to at least one of your classmates' posts. This ensures communication between students. Please contact me if you are unsure of what is expected.



## Week 5 Discussion

Provide a program in the target language of our compiler/interpreter that incorporates all elements of the language. It must include all of the following:

- Integer, Real and Boolean literals
- Every arithmetic operator: + - * / rem **
- Every relational operator: = /= > >= < <=
- Every logical operator: and or not
- An if statement
- A case statement
- A reduce statement
- Multiple variable declarations
- Multiple parameter declarations

In addition, your program should be written in a such a way that it test the associativity and precedence of all operators. It should also include nested statements.

Also provide several sets of input (values for the parameters) to the program, enough so that every line of code within the program is executed at least once. For each input supplied, provide the corresponding output.

# Solution

## Algorithm:

The following program incorporates all elements of the language including integer, real and boolean literals, arithmetic operators (+ - * / rem **), relational operators (= /= > >= ‹ <=), logical operators (and or not), if statement, case statement, reduce statement, multiple variable declarations, and multiple parameter declarations. The program also tests the associativity and precedence of all operators and includes nested statements.

- The program starts with a function called `main` which takes four parameters - `a` (boolean), `b` (integer), `flag` (boolean), and `z` (boolean) and returns a boolean value.
- Inside the function, multiple variable declarations are made including `x` (integer), `y` (real), and `result` (boolean).
- The integer literal value 10 is assigned to `x`, and the real literal value 2.5 is assigned to `y`. The boolean literal value false is assigned to `result`.
- Next, an if statement is used to evaluate the boolean expression `(b > 5) and flag or not (z)`. If the expression evaluates to true, `x` is incremented by `b`, and `y` is multiplied by `b`. If `x` is less than 20 or `y` is greater than 5.0, `result` is assigned the boolean value true. If the boolean expression evaluates to false, `x` is decremented by `a`, and `y` is divided by `b`. If `x` is greater than or equal to 5 and `y` is less than or equal to 1.0, `result` is assigned the boolean value false.
- A case statement is used to evaluate the expression `x rem 4`. If the result is 0, `y` is incremented by 1.0. If the result is 1, `y` is decremented by 1.0. If the result is 2, `y` is multiplied by 2.0. If the result is 3, `y` is divided by 2.0. If none of the above conditions are met, `y` remains unchanged.
- A reduce statement is used to reduce the value of the expression `x + y ** 2 / b`. The reduce operation is multiplication (`*`).
- Finally, the value of `result` is returned from the function.
- To test every line of code within the program, several sets of input values should be provided for the parameters `a`, `b`, `flag`, and `z`. The values should be chosen such that every possible path through the if statement and case statement is executed at least once.

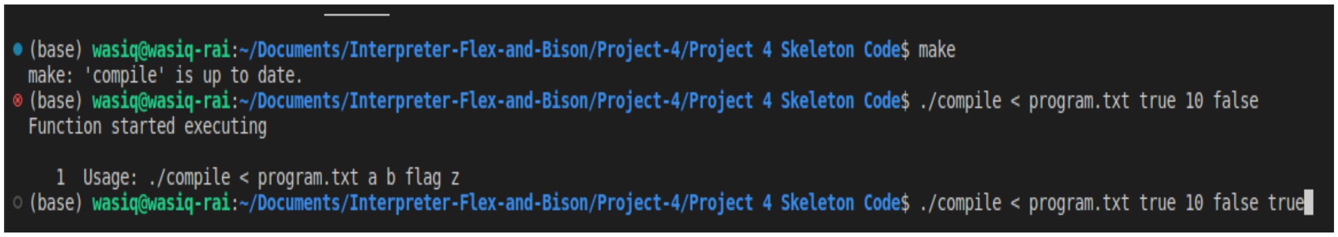
## Code:

```
-- Multiple parameter declarations
-- The function main takes in four parameters, a boolean a, an integer b, a boolean flag, and a boolean z.
function main a: boolean, b: integer, flag: boolean, z: boolean returns boolean;

-- Multiple variable declarations
-- Declare three variables, x as an integer, y as a real, and result as a boolean.
  x: integer ;
  y: real;
  result: boolean;

  begin
    -- Assign values to x, y, and result.
    x = 10;
    y = 2.5;
    result = false;

    -- Test if b is greater than 5 and flag is true, or if z is false.
    if (b > 5) and flag or not (z) then
      -- If true, add b to x and multiply y by b.
      x = x + b;
      y = y * b;

      -- Test if x is less than 20 or if y is greater than 5.0.
      if x < 20 or y > 5.0 then
        -- If true, set result to true.
        result = true;
      endif;
    else
    -- If false, subtract a from x and divide y by b.
    x = x - a;
    y = y / b;

    -- Test if x is greater than or equal to 5 and if y is less than or equal to 1.0.
    if x >= 5 and y <= 1.0 then
        -- If true, set result to false.
        result = false;
    endif;
endif;

-- Use a case statement to check the remainder of x divided by 4.
case x rem 4 is
    -- If the remainder is 0, add 1.0 to y.
    when 0 =>
        y = y + 1.0;
    -- If the remainder is 1, subtract 1.0 from y.
    when 1 =>
        y = y - 1.0;
    -- If the remainder is 2, multiply y by 2.0.
    when 2 =>
        y = y * 2.0;
    -- If the remainder is 3, divide y by 2.0.
    when 3 =>
        y = y / 2.0;
    -- If the remainder is anything else, leave y unchanged.
    others =>
        y = y;
endcase;
```

```
  -- Use a reduce statement to multiply x by the result of y raised to the power of 2 divided by b.
  reduce *
     x + y ** 2 / b ;
  endreduce;

  -- Return the value of result.
  returns  result;
end;
```

## INPUT:

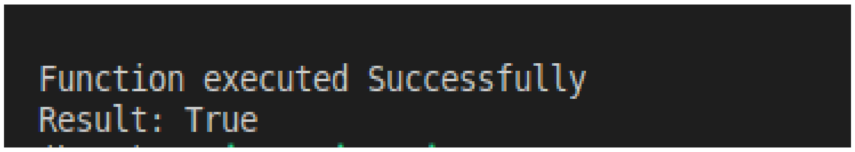
Input is to be given in command prompt. Four variables `a`, `b`, `flag`, and `z` are to be given while compiling the program.



```
● (base) wasiq@wasiq-rai:~/Documents/Interpreter-Flex-and-Bison/Project-4/Project 4 Skeleton Code$ make
  make: 'compile' is up to date.
⊗ (base) wasiq@wasiq-rai:~/Documents/Interpreter-Flex-and-Bison/Project-4/Project 4 Skeleton Code$ ./compile < program.txt true 10 false
  Function started executing

    1  Usage: ./compile < program.txt a b flag z
○ (base) wasiq@wasiq-rai:~/Documents/Interpreter-Flex-and-Bison/Project-4/Project 4 Skeleton Code$ ./compile < program.txt true 10 false true
```

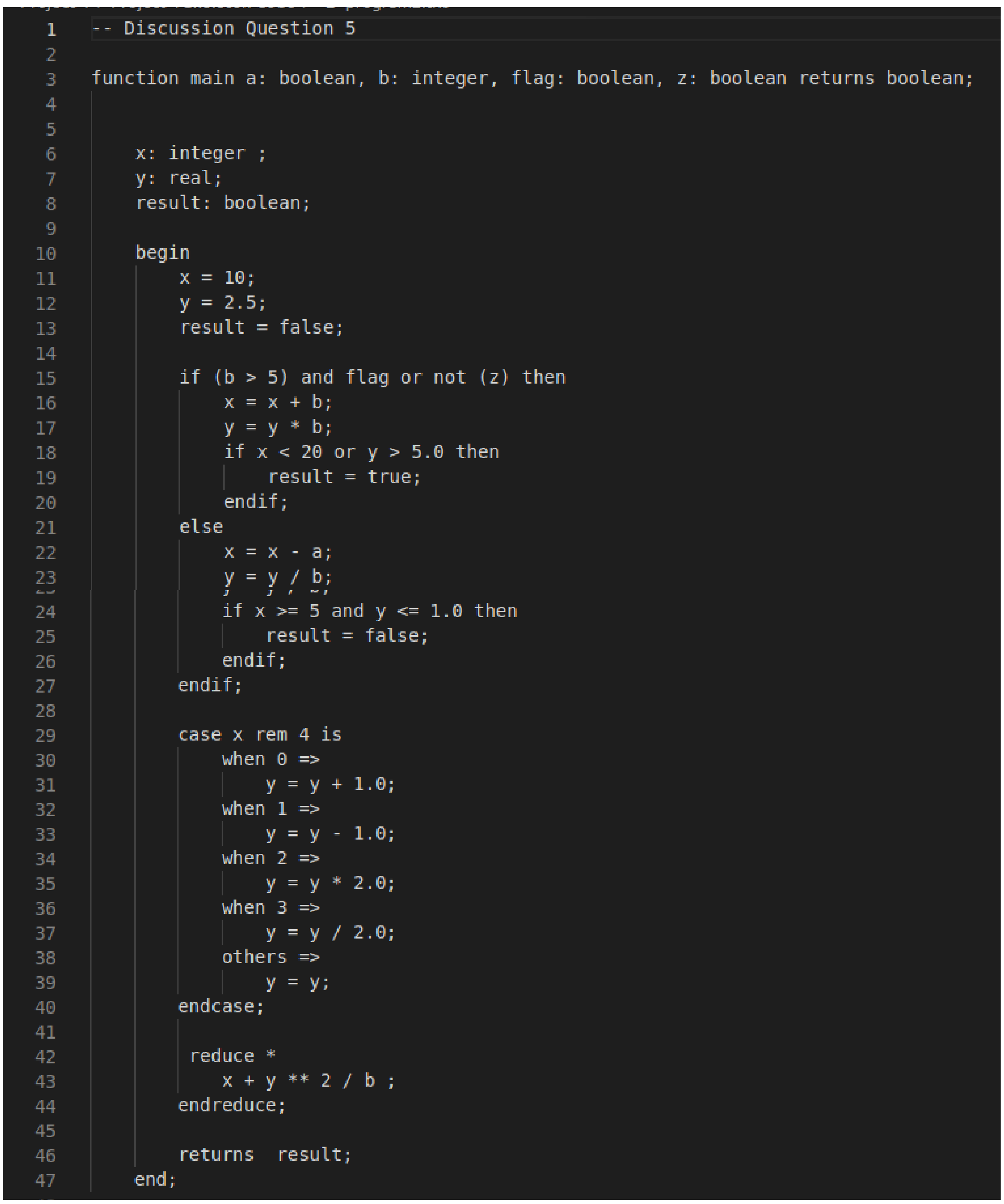
## OUTPUT:

After certain calculations and if else statements, the resultant value will be displayed as result. As our program function Is returning a boolean value, the result will either be true or false.



```
Function executed Successfully
Result: True
```

## SCREENSHOT:



```
-- Discussion Question 5

function main a: boolean, b: integer, flag: boolean, z: boolean returns boolean;



    x: integer ;
    y: real;
    result: boolean;

    begin
        x = 10;
        y = 2.5;
        result = false;

        if (b > 5) and flag or not (z) then
            x = x + b;
            y = y * b;
            if x < 20 or y > 5.0 then
                result = true;
            endif;
        else
            x = x - a;
            y = y / b;
            if x >= 5 and y <= 1.0 then
                result = false;
            endif;
        endif;

        case x rem 4 is
            when 0 =>
                y = y + 1.0;
            when 1 =>
                y = y - 1.0;
            when 2 =>
                y = y * 2.0;
            when 3 =>
                y = y / 2.0;
            others =>
                y = y;
        endcase;

         reduce *
            x + y ** 2 / b ;
        endreduce;

        returns  result;
    end;
```

## SETS OF INPUT VALUES:

Here are some sets of input values and their expected results:

- **Set 1:** a = true b = 10 flag = true z = false
  - Expected result: **true**
- **Set 2:** a = 0 b = 5 flag = 0 z = 1
  - Expected result: **false**
- **Set 3:** a = true b = 3 flag = false z = false
  - Expected result: **false**
- **Set 4:** a = false b = 12 flag = true z = true
  - Expected result: **true**
- **Set 5:** a = true b = 6 flag = true z = true
  - Expected result: **true**

This input parameters are to be given while compiling the program as :

```
./compile < program.txt false 10 true true
```

**or**

```
./compile < program.txt 1 5 0 1
```

## References:

1. "Compilers: Principles, Techniques, and Tools" by Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman
2. "The C Programming Language" by Brian W. Kernighan and Dennis M. Ritchie
3. "Programming Language Pragmatics" by Michael L. Scott
4. "Clean Code: A Handbook of Agile Software Craftsmanship" by Robert C. Martin