Recursive to Iterative

1) The pseudocode given below converts decimal number to corresponding binary.

```
FUNCTION ToBinaryRecursive(Dec):
    IF Dec = 0 THEN
        RETURN 0
    ELSE:
        RETURN ToBinaryRecursive(Dec DIV 2)*10 + (Dec MOD 2)
```

i) Write program code to declare the function ToBinaryRecursive().

ii) Write program code for the main program. The main program needs to run all three of the following function calls and output the result of each call:

- ToBinaryRecursive(7)
- ToBinaryRecursive(0)
- ToBinaryRecursive(10)

iii) Rewrite the function Unknown() as an iterative function, ToBinaryIterative().

iv) The iterative function needs to be called three times with the same parameters as in part (ii). Ammend the main program to perform those calls.

2) The pseudocode given below prints some pattern:

```
FUNCTION PrintPattern(N):
    IF N < 0 THEN:
        PRINT "The pattern broke."
        RETURN
    IF N = 0 THEN
        RETURN 0
    ELSE:
        PRINT N, Pattern(N - 1), N
```

i) Write program code to declare the function *PrintPattern()*.

ii) Write program code for *IterativePattern()* that prints the same patterns.

iii) Call both versions of function with input 5, -1 and 9.

Iterative to Recursive

1) The given pseudocode converts a binary number provided as a string to the string representation of the corresponding denary.

```
FUNCTION ToDenaryIterative(BinaryString):
    Denary <- 0
    Length <- LENGTH(BinaryString)
    FOR i <- 0 TO Length - 1 DO
        Denary <- Denary * 2 + INT(MID(BinaryString, i, i+1))
    ENDFOR
    RETURN Denary
```

i) Write program code defining the function ToDenaryIterative().

ii) Write program code for recursive version ToDenaryRecursive().

iii) Test both versions of function with same value.

2) The given pseudocode calculates the factorial of a number iteratively.

```
FUNCTION FactorialIterative(N):
    Result <- 1
    FOR i <- 1 TO N
        Result <- Result * i
    NEXT i
    RETURN Result
```

i) Write program code defining the function FactorialIterative().

ii) Write program code for recursive version FactorialRecursive().

iii) Test both versions of the function with the same value.

Questions present below are from past papers.

Recursive to Iterative

> Question taken from 2021-Oct-Nov-41 (9608)

1) Given below is a recursively defined function.

```
FUNCTION Recursive(Num1, Num2 : INTEGER) RETURNS INTEGER
    IF Num1 < 0 OR Num2 < 0 THEN
        RETURN 1
    ELSE
        IF Num1 < Num2 THEN
            Num1 <- Num1 - 2
            RETURN 20 + 2 * Recursive(Num1, Num2)
        ELSE
            Num2 <- Num2 - 2
```

```
            RETURN 10 + 2 * Recursive(Num1, Num2)
        ENDIF
    ENDIF
ENDFUNCTION
```

i) Write program corresponding to the above pseudo code.

ii) Call the function with parameters 5 and 6.

iii) Write program that defines function Iterative(Num1, Num2) that works similar to the function above but iteratively.

iv) Call the Iterative() function with parameters 5 and 6.

> Question taken from 2021-Oct-Nov-41

2) Study the following pseudocode for a recursive function:

```
FUNCTION Unknown(BYVAL X, BYVAL Y : INTEGER) RETURNS INTEGER
    IF X < Y THEN
        OUTPUT X + Y
        RETURN (Unknown(X + 1, Y) * 2)
    ELSE
        IF X = Y THEN
            RETURN 1
        ELSE
            OUTPUT X + Y
            RETURN (Unknown(X - 1, Y) DIV 2)
        ENDIF
    ENDIF
ENDFUNCTION
```

i) Write program code to declare the function **Unknown()**.

ii) The main program needs to run all three of the following function calls and output the result of each call. Write program code for the main program.

- Unknown(10, 15)
- Unknown(10, 10)
- Unknown(15, 10)

iii) Rewrite the function **Unknown()** as an iterative function, **IterativeUnknown()**.

iv) Call **IterativeUnknown()** with same parameters as in **ii**.

Iterative to Recursive

> Question taken from 2023-Oct-Nov-42

1) The iterative function IterativeCalculate() totals all the divisors of its integer parameter and returns this total.

```
FUNCTION IterativeCalculate(Number : INTEGER) RETURNS INTEGER
    DECLARE Total : Integer
    DECLARE ToFind : Integer
    ToFind ← Number
    Total ← 0
    WHILE Number <> 0
        IF ToFind MODULUS Number = 0 THEN
            Total ← Total + Number
        ENDIF
        Number ← Number - 1
    ENDWHILE
    RETURN Total
ENDFUNCTION
```

i) Write program code for **IterativeCalculate()**.

ii) Write program code to call **IterativeCalculate()** with 10 as the parameter and output the return value.

iii) Write program code for recursive version **RecursiveValue(Number : INTEGER, ToFind : INTEGER)**.

> Question taken from 2023-Oct-Nov-43

2) This iterative pseudocode algorithm for the function IterativeVowels() takes a string as a parameter and counts the number of lower-case vowels in this string.

```
FUNCTION IterativeVowels(Value : STRING) RETURNS INTEGER
    DECLARE Total : INTEGER
    DECLARE LengthString : INTEGER
    DECLARE FirstCharacter : CHAR
    Total ← 0
    LengthString ← LENGTH(Value)
    FOR X ← 0 TO LengthString - 1
        FirstCharacter ← MID(Value, 0, 1)
        IF FirstCharacter = 'a' OR FirstCharacter = 'e' OR FirstCharacter =
'i' OR FirstCharacter = 'o' OR FirstCharacter = 'u' THEN
            Total ← Total + 1
        ENDIF
        Value ← MID(Value, 1, LENGTH(Value)-1)
    NEXT X
    RETURN Total
ENDFUNCTION
```

i) Write program code for the function **IterativeVowels()**.

ii) Write program code to call the function IterativeVowels() with the parameter "house" from the main program.

iii) Rewrite the function IterativeVowels() as a recursive function with the identifier RecursiveVowels().

iv) Write program code to call the function RecursiveVowels() with the parameter "imagine" from the main program.
