Ch24

Recursion

CHAPTER ASSIGNMENT · 31 MARKS

Name _____ Class _____ No. _____ Date _____

Answer every question in the space given. Show your working. You may use the chapter page to help you.

Questions from *Computer Science for Cambridge International AS & A Level Coursebook*, 2nd edition, Sylvia Langfield and Dave Duddell (Cambridge University Press), for classroom use.

Tasks and questions from the chapter

DP **Discussion Point:** Carefully examine the two solutions to the factorial function. What happens if the iterative function is called with parameter 0? What happens if the recursive function is called with parameter 0? What changes would need to be made so the mathematical definition holds for all values of n?

T24.01 **Task 24.01:** Write program code to implement the recursive algorithm for the factorial function.

Q24.01 **Question 24.01:** What happens when the function is called with `Factorial(-2)`? Which rule is not satisfied?

T24.02 **Task 24.02:** Consider the following recursive algorithm:

```
PROCEDURE X(n : INTEGER)
   IF (n = 0) OR (n = 1)
      THEN
         OUTPUT n
      ELSE
         CALL X(n DIV 2)
         OUTPUT(n MOD 2)
   ENDIF
ENDPROCEDURE
```

Dry-run the procedure call `X(19)` by completing a trace table. What is the purpose of this algorithm?

T24.03 **Task 24.03:** Use your program code from Task 24.01 and add the main program as shown in Worked Example 24.03.

Amend your code in the following ways (line numbers are relative to the pseudocode in Worked Example 24.03):

- Add a global integer variable `CallNumber`
- Initialise `CallNumber` to zero (line 155).
- Increment `CallNumber` (line 35).
- Add a statement to output the values of `CallNumber` and `n` (line 36).
- Add a statement to output the value of `Result` (line 95).

Run the program and study the output.

RP **Reflection Point:** How can you tell from a function or procedure definition whether or not it is recursive?

Exam-style questions

1 (a) Outline the differences between iteration and recursion. [2]

(b) Give **one** advantage and **one** disadvantage of using recursive subroutines. [2]

2 The following is a recursively defined function which calculates the result of $\text{Base}^{\text{Exponent}}$. For example, 2^3 is 8.

```
FUNCTION Power(Base: INTEGER, Exponent : INTEGER) RETURNS INTEGER
   IF Exponent = 0
      THEN
         Result <- 1
      ELSE
         Result <- Base * Power(Base, Exponent - 1)
   ENDIF
   RETURN Result
ENDFUNCTION
```

(a) State what is meant by 'recursively defined'. [1]

(b) Complete a trace table for the execution of the function call `Power(2,4)` showing for each re-entry into the `Power` function, the values passed to the function and the results returned. [6]

(c) Explain the role of the stack in the execution of the `Power` function. [3]

(d) Write a pseudocode non-recursive (iterative) version of the `Power` function. [6]

(i) Give **one** reason why a non-recursive `Power` function may be preferred to a recursive one. [1]

(ii) Give **one** reason why a recursive `Power` function may be preferred to a non-recursive one. [1]

3 The following is a recursively defined function which calculates the nth integer in the sequence of Fibonacci numbers.

```
01  FUNCTION Fibonacci(n : INTEGER) RETURNS INTEGER
02     IF (n = 0) OR (n = 1)
03        THEN
04           Result <- 1
05        ELSE
06           Result <- Fibonacci(n - 1) + Fibonacci(n - 2)
07     ENDIF
08     RETURN Result
09  ENDFUNCTION
```

(i) Give the line number of the line containing the base case. [1]

(ii) Give the line number of the line containing the general case. [1]

(b) Complete a trace table for the function call `Fibonacci(4)`. [7]

End of assignment · Check your answers against the chapter at gurujazzy.com/subjects/computer-science/as-a-level/recursion/