4.1.4 Recursion

Wednesday, 9 December 2020 2:47 PM

4.1.4 Recursion

4.1.4 Recursion

May/June 2011. P21/P22

4. The following pseudocode is a recursive function where *n* is an integer.

```
FUNCTION prod(n)
IF n = 1
  THEN
    prod ← 1
  ELSE
    prod ← n * prod(n-1)
ENDIF
RETURN
```

(a) (i) What value is returned by prod(1)? [1]

(ii) What value is returned by prod(3)? [1]

(b) (i) What happens if the parameter passed is -1? [2]

(ii) What changes will need to be made to the pseudocode to address the problem in (b)(i)? [2]

May/June 2012. P21/22

5 Romana is learning about recursion. She designs a recursive function.

```
01 FUNCTION Happening(Num)
02 IF Num = 1
03   THEN
04     Happening ← 1
05   ELSE
06     Happening ← Happening(Num - 1) + Num
07 ENDIF
08 ENDFUNCTION
```

(a) Calculate the value returned by the function call Happening(4).
Show your working. [6]

(b) (ii) State the line number in the given pseudocode that shows the function is recursive. Give your reason for choosing this. [2]

(c) State what will happen if the function is called with Happening(-1). [1]

4.1.4 Recursion

(d) The same process could have been solved iteratively using a FOR loop.
Discuss the merits of iterative and recursive solutions to this problem, particularly when the initial value of Num is very large. [4]

May/June 2012. P23

5 Ramon is learning about recursion. He has designed a recursive function.

```
FUNCTION Something(Number)
IF Number = 7
  THEN
    Something ← 1
  ELSE
    Something ← Something(Number + 1) + Number
ENDIF
ENDFUNCTION
```

(a) Calculate the value returned by the function call Something(4). Show your working. [6]

(b) State what will happen if the function is called with Something(8). [1]

(c) State what would happen if the fourth line, 'Something ← 1', was not present. [2]

Oct/Nov 2013.P21

4 Juan is also learning about recursion. He writes the pseudocode for a recursive function.

```
1  FUNCTION Add(N)
2    DECLARE R
3      IF N <= 0
4        THEN
5          R ← 0
6        ELSE
7          R ← N + Add(N - 1)
8      ENDIF
9    RETURN R
10 ENDFUNCTION
```

(a) What is the scope of the variable R? [1]
(b) State the line number which shows that this function is recursive. [1]
(c) List the function calls that are generated by an initial call of Add(3). [3]

Oct/Nov 2013.P22

4 Aisha writes the following pseudocode for a recursive function that works out the greatest common divisor of two positive integers:

4.1.4 Recursion

```
0 FUNCTION Divisor(x, y)
1   IF y = 0
2     THEN
3       RETURN x
4     ELSE
5       x ← x MOD y
6       RETURN Divisor(y, x)
7   ENDIF
8 ENDFUNCTION
```

(a) (i) Trace the call Divisor(8, 2).

Line number	x	y	RETURN
0	8	2	

[2]

(ii) Trace the call Divisor(38, 7).

Line number	x	y	RETURN
0	38	7	

[4]

4.1.4 Recursion

Oct/Nov 2013.P23

4 Ashvin is also learning about recursion. He writes the pseudocode for a recursive function.

```
1  FUNCTION Calc(X)
2  DECLARE Temp
3    IF X > 0
4      THEN
5        Temp ← X + Calc(X - 2)
6      ELSE 7 Temp ←
8    ENDIF
9  RETURN Temp
10 ENDFUNCTION
```

(a) What is the scope of the variable Temp? [1]
(b) State the line number which shows that this function is recursive. [1]
(c) List the function calls that are generated by an initial call Calc(5). [3]
(d) What will be returned by the call Calc(6)? [1]

May/June 2014.P21/P22

4 Look at this pseudocode function:

```
FUNCTION Y(s : STRING) RETURNS STRING
  DECLARE x : INTEGER
  x ← LENGTH(s)
  IF x = 1
    THEN
      RETURN s
    ELSE
      RETURN Y(RIGHT(s, x - 1)) + LEFT(s, 1)
      // RIGHT above returns rightmost x - 1 characters of string s
      // LEFT above returns leftmost character of string s
  ENDIF
ENDFUNCTION
```

(i) How can you tell that function Y is recursive? [1]

4.1.4 Recursion

(ii) Dry-run the function when it is called with the string 'BYTE' as parameter.

Call Number	Function call	s	x	RIGHT (s, x - 1)	LEFT (s, 1)	Return value
1	Y ('BYTE')	'BYTE'				

[7]

(iii) What does function Y do? [1]

Computer Science (9608)

May/ June 2015.P43

6 A recursively defined procedure X is defined below:

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

(a) Explain what is meant by recursively defined. [1]
(b) Explain how a stack is used during the execution of a recursive procedure. [2]
(c) Dry run the procedure X by completing the trace table for the procedure call:
CALL X(40)

Call number	n	(n = 0) OR (n = 1)	n DIV 2	n MOD 2
1	40	FALSE	20	
2				
3				
4				
5				
6				

[6]

4.1.4 Recursion

Oct/Nov 2015.P41/P43

4 A binary tree Abstract Data Type (ADT) has these associated operations:

- create the tree (CreateTree)
- add an item to tree (Add)
- output items in ascending order (TraverseTree)

(b) The binary tree ADT is to be implemented as an array of nodes. Each node consists of data and two pointers.

Using pseudocode, a record type, Node, is declared as follows:

```
TYPE Node
   DECLARE Name : STRING
   DECLARE LeftPointer : INTEGER
   DECLARE RightPointer : INTEGER
ENDTYPE
```

The statement

```
DECLARE Tree : ARRAY[1:10] OF Node
```

reserves space for 10 nodes in array Tree.

The CreateTree operation links all nodes into a linked list of free nodes. It also initialises the RootPointer and FreePointer.

Show the contents of the Tree array and the values of the two pointers, RootPointer and FreePointer, after the operations given in **part (a)** have been carried out.

RootPointer: ______

FreePointer: ______

Tree

	Name	LeftPointer	RightPointer
[1]			
[2]			
[3]			
[4]			
[5]			
[6]			
[7]			
[8]			
[9]			
[10]			

[7]

4.1.4 Recursion

(c) A programmer needs an algorithm for outputting items in ascending order. To design this, the programmer writes a recursive procedure in pseudocode.

(i) Complete the pseudocode:

```
01 PROCEDURE TraverseTree(BYVALUE Root: INTEGER)
02   IF Tree[Root].LeftPointer ..............................
03     THEN
04       TraverseTree(..............................)
05   ENDIF
06   OUTPUT..............................................Name
07   IF .......................................... <> 0
08     THEN
09       TraverseTree(..............................)
10   ENDIF
11 ENDPROCEDURE
```

[5]

(ii) Explain what is meant by a recursive procedure. Give a line number from the code above that shows procedure TraverseTree is recursive. [2]

(iii) Write the pseudocode call required to output all names stored in Tree. [1]