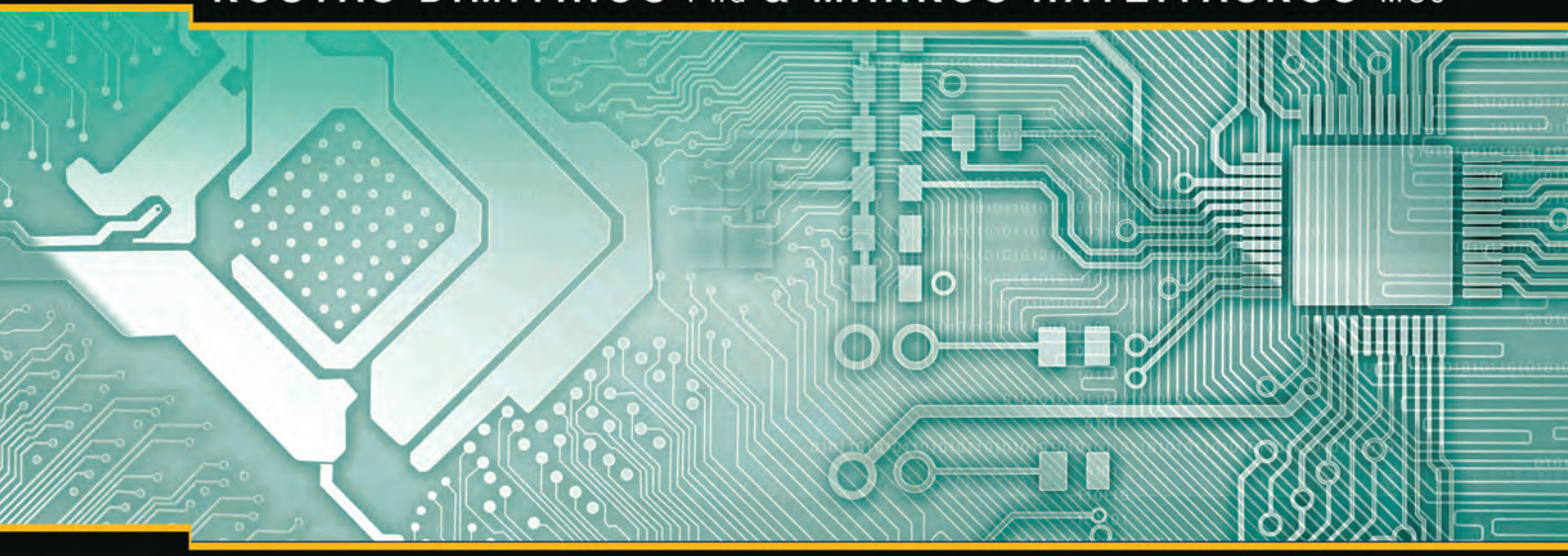
KOSTAS DIMITRIOU Phd & MARKOS HATZITASKOS MSc



ADVANCED COMPUTER SCIENCE

For the IB Diploma Program
(International Baccalaureate)

HIGH LEVEL COMPUTER SCIENCE

Express Publishing

KOSTAS DIMITRIOU Phd & MARKOS HATZITASKOS MSc

ADVANCED COMPUTER SCIENCE

For the IB Diploma Program

(International Baccalaureate)

HIGH LEVEL COMPUTER SCIENCE

Express Publishing

Published by Express Publishing

Liberty House, Greenham Business Park, Newbury,
Berkshire RG19 6HW, United Kingdom
Tel.: (0044) 1635 817 363
Fax: (0044) 1635 817 463
email: inquiries@expresspublishing.co.uk
www.expresspublishing.co.uk

© Express Publishing, 2016

Design and Illustration © Express Publishing, 2016

First published 2016

Made in EU

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form, or by any means, electronic, photocopying, or otherwise, without the prior written permission of the publishers.

This book is not meant to be changed in any way.

ISBN 978-1-4715-5233-5

Copyright page

List of licensed IB material used:

DP Computer Science Guide (first exams 2014):

Topic 5—Abstract data structures
5.1 Abstract data structures
Thinking recursively
5.1.1 Identify a situation that requires the use of recursive thinking.
5.1.2 Identify recursive thinking in a specified problem solution.
5.1.3 Trace a recursive algorithm to express a solution to a problem.
Abstract data structures
5.1.4 Describe the characteristics of a two-dimensional array.
5.1.5 Construct algorithms using two-dimensional arrays.
5.1.6 Describe the characteristics and applications of a stack.
5.1.7 Construct algorithms using the access methods of a stack.
5.1.8 Describe the characteristics and applications of a queue.
5.1.9 Construct algorithms using the access methods of a queue.
5.1.10 Explain the use of arrays as static stacks and queues.
Linked lists
Linked lists will be examined at the level of diagrams and descriptions. Students are not expected to construct linked list algorithms using pseudocode.
5.1.11 Describe the features and characteristics of a dynamic data structure.
5.1.12 Describe how linked lists operate logically.
5.1.13 Sketch linked lists (single, double and circular).
Trees
Binary trees will be examined at the level of diagrams and descriptions. Students are not expected to construct tree algorithms using pseudocode. Tracing and constructing algorithms are not expected.
5.1.14 Describe how trees operate logically (both binary and non-binary).
5.1.15 Define the terms: parent, left-child, right-child, subtree, root and leaf.
5.1.16 State the result of inorder, postorder and preorder tree traversal.
5.1.17 Sketch binary trees.
Applications
5.1.18 Define the term dynamic data structure.
5.1.19 Compare the use of static and dynamic data structures.
5.1.20 Suggest a suitable structure for a given situation.

Topic 6—Resource management
6.1 Resource management
System resources
6.1.1 Identify the resources that need to be managed within a computer system.
6.1.2 Evaluate the resources available in a variety of computer systems.
6.1.3 Identify the limitations of a range of resources in a specified computer system.
6.1.4 Describe the possible problems resulting from the limitations in the resources in a computer system.
Role of the operating system
6.1.5 Explain the role of the operating system in terms of managing memory, peripherals and hardware interfaces.
6.1.7 Outline OS resource management techniques: scheduling, policies, multitasking, virtual memory, paging, interrupt, polling.
6.1.8 Discuss the advantages of producing a dedicated operating system for a device.
6.1.9 Outline how an operating system hides the complexity of the hardware from users and applications.

Topic 7—Control
7.1 Control
Centralized control systems
7.1.1 Discuss a range of control systems.
7.1.2 Outline the uses of microprocessors and sensor input in control systems.
7.1.3 Evaluate different input devices for the collection of data in specified situations.
7.1.4 Explain the relationship between a sensor, the processor and an output transducer.

7.1.5 Describe the role of feedback in a control system.
7.1.6 Discuss the social impacts and ethical considerations associated with the use of embedded systems.

Distributed systems

7.1.7 Compare a centrally controlled system with a distributed system.
7.1.8 Outline the role of autonomous agents acting within a larger system.

D—Object-oriented programming

D.4 Advanced program development

D.4.1 Define the term recursion.
D.4.2 Describe the application of recursive algorithms.
D.4.3 Construct algorithms that use recursion.
D.4.4 Trace recursive algorithms.
D.4.5 Define the term object reference.
D.4.6 Construct algorithms that use reference mechanisms.
D.4.7 Identify the features of the abstract data type (ADT) list.
D.4.8 Describe applications of lists.
D.4.9 Construct algorithms using a static implementation of a list.
D.4.10 Construct list algorithms using object references.
D.4.11 Construct algorithms using the standard library collections included in JETS.
D.4.12 Trace algorithms using the implementations described in assessment statements D.4.9–D.4.11.
D.4.13 Explain the advantages of using library collections.
D.4.14 Outline the features of ADT's stack, queue and binary tree.
D.4.15 Explain the importance of style and naming conventions in code.

KOSTAS DIMITRIOU Phd & MARKOS HATZITASKOS MSc

ADVANCED COMPUTER SCIENCE

For the IB Diploma Program

(International Baccalaureate)

HIGH LEVEL COMPUTER SCIENCE

Express Publishing

Kostas Dimitrou Dedication

To my son Dimitris and my daughter Eliana.

Never forget that I love you.

Markos Hatzitaskos Dedication

To all my friends that helped make this book a reality.

You know who you are.

Preface

Kostas Dimitriou holds a PhD in Spatial Decision Support Systems and Environmental Planning, and has taught computer science courses in various undergraduate and postgraduate University courses. He has participated in many scientific conferences and workshops, twenty research projects, and presented sixty scientific articles. He has been teaching the IB computer science in the Hellenic American Educational Foundation since 2002. He is a Microsoft Certified Educator**,** Microsoft Expert Educator**,** Microsoft Expert Education Trainer and Microsoft Innovative Educator Fellow. {kdimitriou@haef.gr}

Markos Hatzitaskos holds an MSc in Advanced Computing and has taught computer science courses throughout all school levels (from primary to high school and the I.B.). He has been teaching in the Hellenic American Educational Foundation since 2011. In his free time, whenever that might be, he develops mobile applications and attends the Athens School of Fine Arts as an undergraduate. {markosh@haef.gr}

The authors would like to thank the Board of Directors and the Administration of the Hellenic American Educational Foundation (HAEF) for providing an ideal working environment. Thanks are also due to Kostas Ziogas who gave us some valuable advice. Both authors would like to express their gratitude to the employees of Express Publishing and especially to our friend Tzeni Vlachou.

The authors would like to acknowledge the ongoing, valuable support of Sophia Arditzoglou, HAEF IB coordinator, over the years. A lot of Computer Science students contributed with valuable ideas, comments and suggestions on early drafts. The computer science class of 2014 encouraged us to start this book.

The purpose of this document is to facilitate learning and help our colleagues and CS students around the world. This book is based on the IB computer science syllabus and follows the IB computer science syllabus. The authors did their very best to cite all resources used. If you find a source that is not properly cited please report it to the authors. This book was inspired by the book[1]: Jones, R & A. Meyenn. (2004). Computer science Java Enabled. International Baccalaureate. Series, IBID press, Victoria.

[1] Jones, R & Meyenn, A. (2004). Computer science Java enabled. International Baccalaureate. Series, IBID press, Victoria.

The following IBO documents were used during the development of this book:

1. International Baccalaureate Organization. (2004). IBDP Computer Science Guide.
2. International Baccalaureate Organization. (2012). IBDP Computer Science Guide.
3. International Baccalaureate Organization. (2012). IBDP Approved notations for developing pseudocode.
4. International Baccalaureate Organization. (2012). IBDP Java Examination Tool Subset.
5. International Baccalaureate Organization. (2012). IBDP Pseudocode in examinations.

Table of Contents

TOPIC 5 – ABSTRACT DATA STRUCTURES

Most IB compatible pseudocode examples of this book have been tested using the EZ Pcode practice tool found at:

https://dl.dropboxusercontent.com/u/275979/ibcomp/pseduocode/pcode.html

This excellent tool was developed by Mr. Dave Mulkey. The authors wish to express their gratitude to the developer of this valuable educational resource.

© IBO 2012

Topic 5 — Abstract data structures[1]

5.1 Abstract data structures

Thinking recursively

5.1.1 – 5.1.3 Recursive thinking

Exit skills. Students should be able to[1]:
Identify a situation that requires the use of recursive thinking. Identify recursive thinking in a specified problem solution. Trace a recursive algorithm to express a solution to a problem.

Image 5.1: The Towers of Hanoi game

Recursion is when a method calls itself until some terminating condition is met. This is accomplished **without** any specific repetition construct, such as a **while** or a **for** loop. Recursion follows one of the basic problem solving techniques, which is to break down the problem at hand into smaller subtasks. Any algorithm that may be presented in a recursive manner can also be presented in an iterative manner and vice versa. In most cases, recursive algorithms are considered as harder to code.

Towers of Hanoi[2]

In order to gain a firm understanding of the basic idea, as well as the application of recursion, the following example

[1] International Baccalaureate Organization. (2012). IBDP Computer Science Guide.

[2] Towers of Hanoi. (2015, November 17). In *Wikipedia, The Free Encyclopedia*. Retrieved 14:03, November 17, 2014, from https://en.wikipedia.org/wiki/Tower_of_Hanoi

presents what is known as the **Towers of Hanoi**. The Towers of Hanoi is a puzzle that consists of three rods and a number of discs of different sizes, which can slide onto any rod. The puzzle starts with the discs in a neat stack in ascending order of size on the first rod, the smallest at the top, as shown in Image 5.1. The goal of the puzzle is to move the stack of discs from the first rod to the third rod, obeying the following rules:

- A disc may not be placed on top of a smaller one.
- Only one disc may move on every move.
- A disc may not be moved if it is not the top disc on a stack.
- For temporary storage, the third rod may be used.

There are various approaches that can solve the Towers of Hanoi problem, including both iterative and recursive solutions. We will be concentrating on a recursive solution, by recognizing that this puzzle may be solved by breaking it into smaller and smaller similar puzzles, until a solution is reached.

Assume that the rods are named A, B and C and that *n* represents the number of discs (with 1 being the smallest, at the top, and n being the largest, at the bottom). A recursive solution to the Tower of Hanoi problem, in order to move *n* discs from rod A to rod C could be the following:

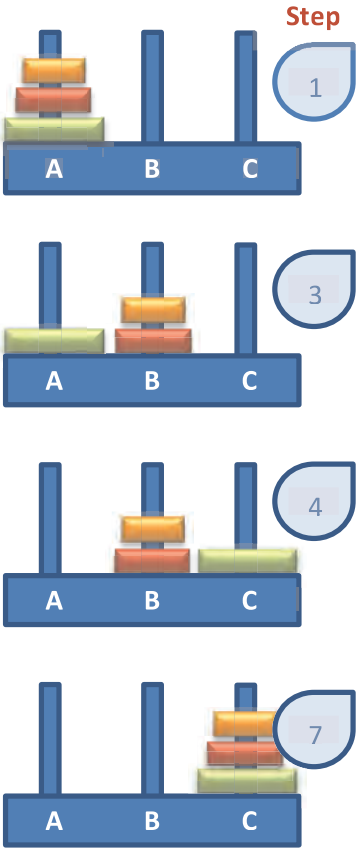


Figure 5.1: Steps of the game

- Move *n-1* discs from rod A to rod B, leaving disc *n* in rod A.
- Move disc *n* from rod A to rod C.
- Move *n-1* discs from rod B to rod C.

The algorithm above is recursive as it is applied again and again in both the first and the third steps for *n-1* discs. At some point *n* will be equal to 1 and a single disc will be moved from rod A to rod C, resulting in an algorithm with finite number of steps.

A working example of this algorithm is examined. Figure 5.1 represents the three rods (named A, B and C) as well as three discs, stacked on top of each other in rod A. The algorithm goes as follows:

1. Move green disc from A to C.
2. Move orange disk from A to B.
3. Move green disk from C to B
4. Move grey disk from A to C
5. Move green disk from B to A
6. Move orange disk from B to C
7. Move green disc from A to C

The recursive algorithm for the solution of the Towers of Hanoi problem is also presented in Figure 5.2. Pay attention to the fact that a sub-procedure called moveDiscs is used. moveDiscs takes four

arguments. The number of the discs (n), the rod the discs are to be moved from (from), the rod to which the discs are to be moved to (dest), as well as the rod that will not be used (aux). The arguments of the moveDiscs sub-procedure (that is, n, from, aux, dest) should not be confused with the name of the rods used previously (A, B and C).

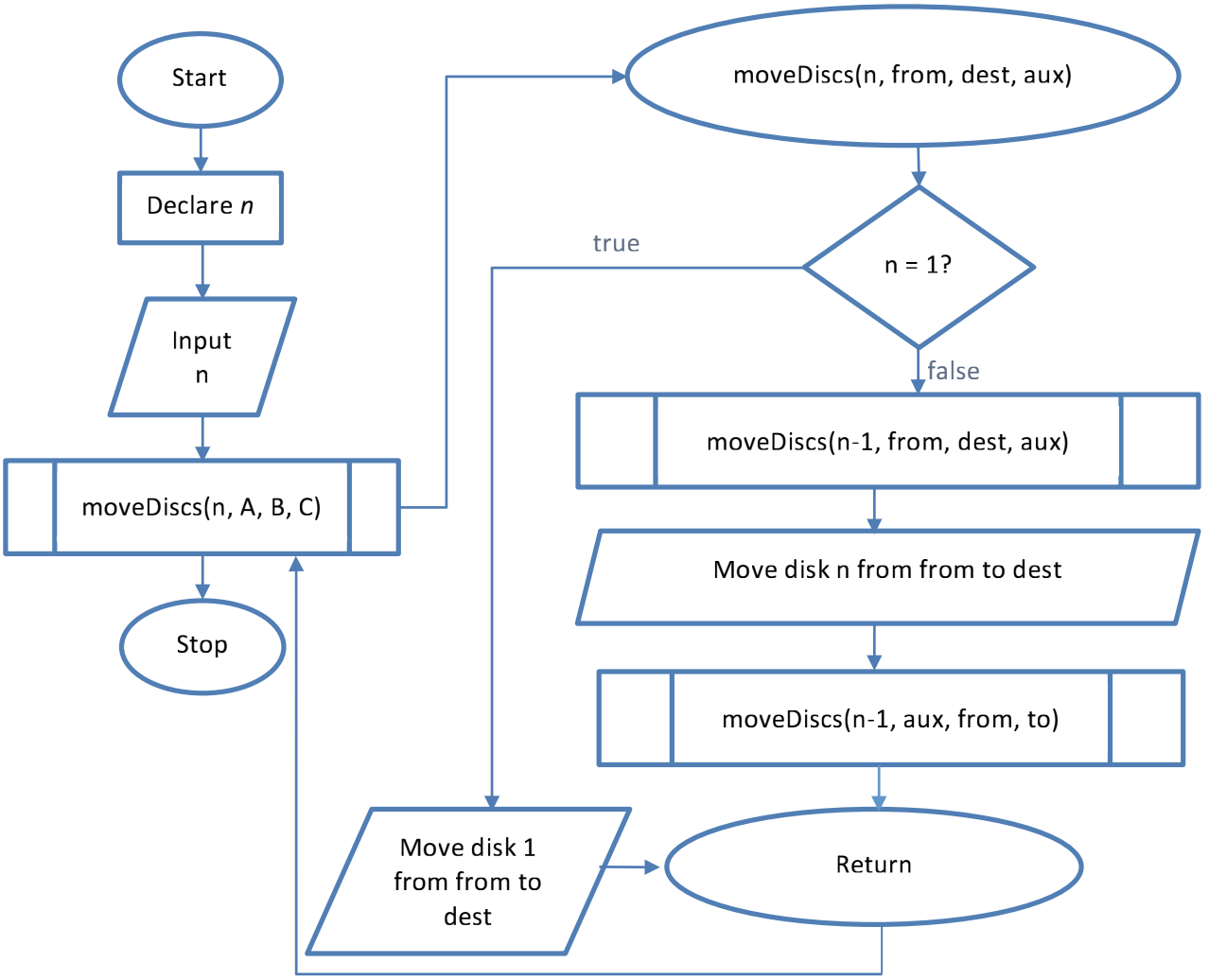


Figure 5.2: The Towers of Hanoi flowchart

Snowflakes

The Koch snowflake is a mathematical curve which is based on the Koch curve, developed by the Swedish mathematician Helge von Koch.

This mathematical curve can be constructed by starting with an equilateral triangle. Using recursion each line segment changes using the following steps:

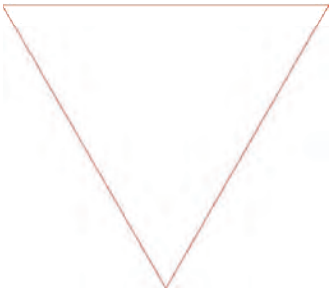
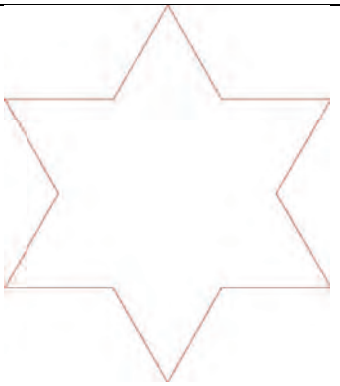
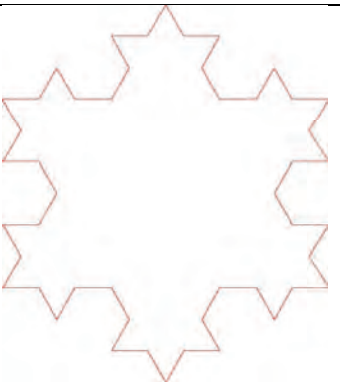
1. divide the initial line segment into three sub-segments of the same length.
2. draw an outward pointing equilateral triangle that has the middle segment from step (1) as its base.
3. delete the line segment that is the base of the triangle from previous step.

The following algorithm expressed in IB pseudocode creates a 400 by 461 window and draws a Koch fractal:

```
 //Three curves that shape an equilateral triangle
 //pen originally is heading at 90 degrees (x axis)
 //the method pen.goForward is supposed to control
 //a pen that plots line segments on the screen
 //the method pen.turnLeft is supposed to change the original
 //heading of the pen counter clockwise by the degrees given as a parameter.
 //the method pen.turnRight is supposed to change the original
 //heading of the pen clockwise by the degrees given as a parameter.
method Draw_Koch_fractal(N)
  width = 400//width of the window
  height = 2*width/Math.sqrt(3)//calculation of the height of the window
  size = width/Math.pow(3.0, N)//size of each drawing step
  initial_pen_position = pen.setposition(0, width*Math.sqrt(3)/2, 0)
 //calculation of the initial pen position (0,114)
  pen.setWindowSize(width, height)//initialization of the window
  koch_fractal(N)//call of the Koch_fractal method
  pen.turnrRight(120)//turn right by 120 degrees
  koch_fractal(N)//call of the Koch_fractal method
  pen.turnRight(120)
  koch_fractal(N)
end method

method koch_fractal(n)
   if (n == 0) then
      pen.goForward(size)
   else
      koch_fractal(n-1)
      pen.turnLeft(60)
      koch_fractal(n-1)
      pen.turnRight(120)
      koch_fractal(n-1)
      pen.turnLeft(60)
      koch_fractal(n-1)
    end if
end method
output Draw_Koch_fractal(N)
```

The following table depicts the snowflakes produced by the above algorithm for N=0 to 5:

 N=0	 N=1	 N=2

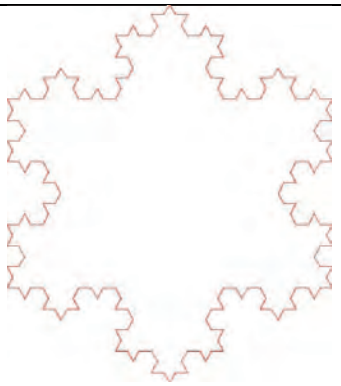
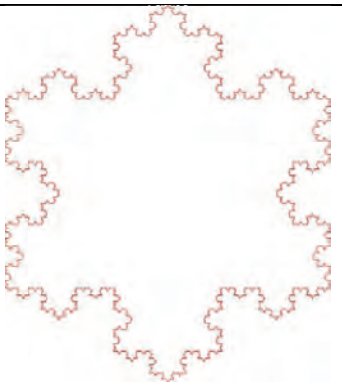
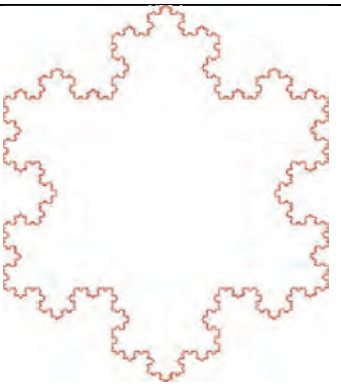
 N=3	 N=4	 N=5

Table 5.1: Various fractals

Programming Example 1: Simple recursion – adding integers.

The following program uses recursion to create the method `addIntUpTo(n)` for `n>0` that will add all numbers from and including `n` down to 1. For example, if `addIntUpTo(4)` is called, the result would be: `4 + 3 + 2 + 1 = 10`

```
method addIntUpTo(n)
     if (n == 1) then
          return 1
     else
          return n + addIntUpTo(n-1)
     end if
end method
```

This method is a recursive function since it calls itself. On each call, the argument is reduced by one (every time `addIntUpTo` is called, its argument is `n-1`). `n-1` calls are made until the terminating condition `n = 1` is met.

Programming Example 2: Example of recursion.

What is going to be the output of the following algorithm?

```
method foo(n)
  if (n <= 1) then
       return 1
  else
       return foo(n-1) + foo(n-2)
  end if
end method

output foo(5)
```

Answer: **8**

Programming Example 3: Example of recursion.

What is going to be the output of the following algorithm?

```
method foo(n, m)
  if (n <= 1) OR (m <= 1) then
       return 2
```

```
  else
      return foo(n-1, m) + foo(n, m-2)
  end if
end method

output foo(5,4)
```

Answer: **30**

Programming Example 4: Example of recursion.

What is going to be the output of the following algorithm?

```
method foo(n, m)
  output "value of n=", n, "value of m =", m
  if (n <= 1) OR (m<=1) then
      return 2
  else
      return foo(n-1, m-n)+foo(n, m-2)
  end if
end method

output "Output is", foo(3,2)
```

Answer:

```
value of n= 3 value of m = 2
value of n= 2 value of m = -1
value of n= 3 value of m = 0
Output is 4
```

Programming Example 5: Example of recursion.

What is going to be the output of the following algorithm?

```
method Foo(X,Y)
 if X < Y then
  return Foo(X+1,Y-2)
 else if X = Y then
  return 2*Foo(X+2,Y-3)-3
 else
  return 2*X+3*Y
 end if
end method
output "Output is", Foo(3,12)
```

Answer:

```
Output is 47
```

ADVANCED COMPUTER SCIENCE

For the IB Diploma Program (International Baccalaureate) HIGH LEVEL COMPUTER SCIENCE

Advanced Computer Science: For the IB Diploma Program is a new educational resource for all students who need to understand the High Level topics of Computer Science. This book references all the assessment statements in the 2014 IB Computer Science subject guide, while remaining flexible enough to be used in any educational setting, including programming courses of moderate to advanced difficulty.

The book uses plain English allowing native and non-native young learners to master the IB computer science High Level course. Advanced computer science for the IB Diploma Program builds upon the successful Core Computer Science book and facilitates High Level students to master the necessary topics.

Feature include:

- Plain English language
- Diagrams and illustrations for all key concepts
- Examination style questions with answers for all topics
- Use of IB style pseudocode, as well as actual working code in the Java Programming Language
- A complete and efficient working Java solution that addresses a real life scenario
- One-to-one reference to the assessment statements in the IB Computer Science subject guide

This book allows both students and teachers to follow the wide-ranging IB Computer Science syllabus in such a way that they can be confident that all aspects of theory and practical exercises have been covered.

Further details about the book, including errata and programming code, can be obtained at the following website: *http://www.expresspublishing.co.uk/ibadvancedcomputerscience*

Express Publishing

ISBN 978-1-4715-5233-5