Pseudocode Answers

Wednesday, 21 September 2011
9:28 AM

Q7 2006 Exam

Question 7

One of the errors that Peter has found is in the code to control the shade curtain motor.
For her flowering orchids, Lucy wants the shade curtain to be closed if the light exceeds 15 000 lux (MaxLight). She also wants the shade curtain closed when the temperature exceeds 33°C (MaxTemp). The curtain should only be opened when the light falls below 7 000 lux (MinLight) and when the temperature drops below 30°C (MinTemp). The light and temperature values will vary depending on the plants grown in each greenhouse 'room'.
Peter has found that the code follows the algorithm below.

```
Begin
    If Temperature > MaxTemp Or Light >MaxLight Then
        CurtainMotor (Close)
    Endif
    If Temperature < MinTemp Or Light < MinLight Then
        CurtainMotor (Open)
    Endif
End
```

a. Below is a testing table. Work out the action that the algorithm finally produces and complete the **last** column.

		Action required	Action produced by algorithm
temperature	**light**	**curtain motor**	**curtain motor**
34	16000	Close	close
34	5000	Close	**(close, then) OPEN***
29	5000	Open	open
29	16000	Close	(close, then) open

4 marks

**13 Nov 06 - my original answer to the second test case was "Close" based on the action of the first IF statement. Thanks to clever Hon for correcting me on it: it should be "Open". I wonder if the examiners noticed that too.*

*If you look at the algorithm, the t=34 triggers the **first** IF statement (which closes the curtain). But then in the **second** IF statement the L=5000 triggers the curtain **opening**. So the curtain closes, then opens, so the algorithm **finally** produces an OPEN status - which is what the question asked. Ooops!*

Also, in test case 4 the curtain also does an unnecessary closure before (erroneously) opening.

b. A line in the algorithm contains a logic error. Identify that line by writing it out in full. 1 mark

If Temperature < MinTemp **Or** Light < MinLight **Then**

c. Describe how this error can be corrected. 1 mark

Change it to

If Temperature < MinTemp **AND** Light < MinLight **Then**

*Fortunately this patch fixes **both** of the cases of algorithm misbehaviour!*

Q3 - 2007 Exam

Kayla has completed her analysis of the system. She has decided to start the design of the new system with the design of the software. From her analysis she knows that one thing the system has to do is to calculate the required amount of each ingredient.

She designs an algorithm that

- reads a file to get the total number of different ingredients Sebastian keeps in stock (Num_Ingredients)
- for each bread item (Product_ID) ordered, uses recipe data and the number of items ordered (Num _ Ord) to calculate the amount of each ingredient required for that item (Amount_ Req)
- adds the ingredient amounts required for each item to get the total quantity needed for each ingredient Qty ()

The procedure will have passed to it data that lists the amount of each ingredient required for every bread item made by the bakery (Product_Recipes)

Here is Kayla's algorithm.

```
PROCEDURE Calc_Qty(Product_Recipes)
BEGIN
    Open File
    Ingredient_ID ← 1
    READ Num_Ingredients
    REPEAT
        Qty(Ingredient_ID) ← 0
        Ingredient_ID ← Ingredient_ID + 1
    UNTIL Ingredient_ID > Num_Ingredients
    REPEAT
        READ Product_ID, Num_Ord
        Ingredient_ID ← 1
        REPEAT
            Ingredient_ID ←Ingredient_ID + 1
            Amount_Req ← Num_Ord * Product_Recipes(Product_ID, Ingredient_ID)
            Qty(Ingredient_ID) ← Qty(Ingredient_ID) + Amount_Req
        UNTIL Ingredient_ID = Num_Ingredients
    UNTIL End Of File
    Close File
END
```

Question 3

a. What is the purpose of the **first** REPEAT - UNTIL loop? (1 mark) State average=0.1

It initialises all of the Qty array's elements to zero. It uses Ingredient_ID as the loop's index (counter).

The REPEAT – UNTIL loop initialises (sets up) the array to store the quantity of each ingredient.

To test the algorithm Kayla decides to use some simplified test data with only 1 bread product and only 3 ingredients.

b. Using the data below complete the test table (3 marks) State average=0.3

Test data

Variable	**Initial Value**
Num_Ingredients	3
Product ID	1
Num Ordered	10
Product_ Recipes(1,1)	0.15
Product_ Recipes(2, 1)	0
Product_Recipes(3,1)	0.20

Test table

Variable	Expected value	Actual value
Qty(1)	1.5	0.0
Qty(2)	0.0	0.0
Qty(3)	2.0	2.0

c. Describe the error in this algorithm (1 mark) State average=0.1

The initialising of Ingredient_ID to 1 in line 12 is wrong because the variable is incremented at the start of the loop, so the starting value is actually 2, not 1. This makes it skip the first ingredient altogether.

Ingredient_ID is set to 1 in the first of the nested repeat loops and immediately in the second part of the nested loop it is increased by 1, so the amount of the first ingredient is never calculated.
Students needed to have a full understanding of each of the elements within the algorithm in order to answer this part correctly.

d. Suggest **one** way the algorithm could be altered to fix this error (2 marks) State average=0.15

Line 12 should be...

Ingredient_ID <- 0

Pretend <- is a left-facing arrow. Heaven knows why they use that symbol in exams - it's hard to put in a webpage!

Either of:
• change the initial value of Ingredient_ID from 1 to 0 in the second loop
• move the Ingredient_ID increment line to the end of the loop and change the terminal conditions from UNTIL Ingredient_ID = Num_Ingredients to UNTIL Ingredient_ID > Num_Ingredients
Question 3 challenged students. Only a very small number completed all parts successfully, and over 80 per cent scored no marks for all parts. The algorithm in the question was not overly complicated; however, the use of a two dimensional array may have provided an extra element of difficulty that prevented many students from attempting the question.

Question C5 - 2008

Question 5

The software specifications show that a procedure is required to enable Pattie to see which event coordinators are available on a particular date. The algorithm for this procedure has been created and is shown below.

The algorithm uses the following variables.

Variable	Use
DateRequired	Stores the date Pattie requires a coordinator
Num_Coordinators	Stores the number of coordinators employed by Pattie
CountCoordinator	Counter for coordinators
Coordinator_Available[]	Stores the availability (true/false) of each coordinator
CoordinatorBooking	Stores a booking date for a coordinator
Coordinator_Name[]	Stores the names of the coordinators

```
BEGIN
  /* initialise data */
  INPUT Date_Required
  READ Num_Coordinators      /* from the Bookings file */

  Set_Up_Coord_Info          /* a procedure that sets all Coordinator_Available[] to True */
                             /* and reads all names into Coordinator_Name[]  */
  Count_Coordinator ← 1

  REPEAT
    Count_Coordinator ← Count_Coordinator + 1
```

```
    /* Read dates already booked for a Coordinator one at a time from the Bookings file */
    REPEAT
        READ Coordinator_Booking
        IF Date_Required = Coordinator_Booking THEN
            Coordinator_Available[Count_Coordinator] ← False
        ENDIF
    UNTIL Coordinator_Booking = Blank

    /* Print Coordinator_Name if he/she is available */
    IF Coordinator_Available[Count_Coordinator] = True THEN
        PRINT Coordinator_Name[Count_Coordinator]
    ENDIF

  UNTIL Count_Coordinator = Num_Coordinators
END
```

Test data used to check algorithm

Variable	**Test data**
Date_Required	13/11/2008
Coordinator_Name[1]	Pattie
Coordinator_Name[2]	Charles
Coordinator_Name[3]	Manfred

Bookings file

3
2/5/2008
12/12/2008
Blank
13/11/2008
15/11/2008
Blank
2/11/2008
12/11/2008
Blank

a. Using the test data above

i. What output would you expect to get if the algorithm was correct?

Pattie

Manfred

ii. What output does the algorithm actually give?

Charles

2 marks - average 0.3

b. Describe the error in the algorithm.

It sets the count_coordinator to 1 during initialisation and does it again as soon as the outside REPEAT begins, so the starting count is 1 higher than it should be.
Therefore only the first two coordinators are tested, and it sets the wrong coordinator's availability flag.

2 marks - average 0.5

The Count_Coordinator was initialised at 1 and then immediately increased to 2 before the first coordinator's details were accessed. Therefore, the first coordinator's details were allocated to the second coordinator, and the second to the third, without the third ever being accessed. Many students recognised this error even if they could not complete part a.

c. Suggest one way the algorithm could be altered to fix this error.

Change the initialisation to Count_coordinators = 0

or (not as easily)...

Move the count increment to just before the UNTIL count_coordinator = num_coordinators, and change the '=' to '>'

2 marks - average 0.5

To fix the error initialise the Count_Coordinator to 0 instead of 1.
An alternative patch would be to move the line 'Count_Coordinator <-- Count Coordinator + 1' to just before the line 'UNTIL Count_Coordinator = Num_Coordinators'.
If the second method was chosen, students needed to alter the UNTIL line by changing the '=' sign to a '>' sign or changing Num_Coordinators to Num_Coordinators + 1.
Students who answered Question 5b. correctly generally also answered this question correctly. However, students who chose the second method of fixing the error generally did not mention the change to the UNTIL line and only obtained half marks.

This question was presented in two parts. The first part required students to read the test data and the description of what was wanted and predict what the algorithm should have produced.
The second part of the question required students to desk check the algorithm to find out what was produced.
In responses where this was done well, the answers to parts b and c of the question were also done well.
Students should be able to create expected results from test data and also desk check algorithms, although most did not do so.

Let's walk through the pseudocode with the test data.. Be careful with these questions because they are usually studded with very subtle "gotchas" that trip up the unwary or casual reader. A walkthrough takes some time, but it's worth it. I thought after a quick scan taht I had easily discovered the gotcha - until I found that a second gotcha got me. Only the walkthrough uncovered it.

- *BEGIN*
- *Input Date_required (13/11/2008)*
- *Read Num_coordinators (3)*
- *Set_Up_coord_Info - so*
- *Coordinator_available[1]=TRUE*
- *Coordinator_available[2]=TRUE*
- *Coordinator_available[3]=TRUE*
- *Coordinator_name[1]=Pattie*
- *Coordinator_name[2]=Charles*
- *Coordinator_name[3]=Manfred*
- *Count_coordinator == 1 (this causes the logical error later)*
- *REPEAT*
- *Count_coordinator ==2*
- *REPEAT*
- *Read coordinator_booking (2/5/08)*
- *Date_required (13/11/2008) <> coordinator_booking (2/5/08) so do nothing*
- *Read coordinator_booking (12/12/08)*
- *Date_required (13/11/2008) <> coordinator_booking (12/12/08) so do nothing*
- *Read coordinator_booking (Blank)*
- *Drop through the UNTIL because coordinator_booking == Blank*
- *Coordinator_available[2] still = TRUE so print Coordinator_name[2] which is **Charles***
- *Count_coordinator ==2 <> Num_Coordinators (3) so return to the first repeat*
- *REPEAT*
- *Count_coordinator ==3*
- *REPEAT*
- *Read coordinator_booking (13/11/2008)*
- *Date_required (13/11/2008) == coordinator_booking (13/11/08) so Coordinator_available[count_coordinator, which is 3] == FALSE*
- *Read coordinator_booking (15/11/08)*
- *Date_required (13/11/2008) == coordinator_booking (13/11/08) so Coordinator_available[count_coordinator, which is 3] == FALSE*
- *Read coordinator_booking (15/11/08)*
- *Date_required (13/11/2008) <> coordinator_booking (15/11/08) so do nothing*
- *Read coordinator_booking (Blank)*
- *Drop through the UNTIL because coordinator_booking == Blank*
- *coordinator_available[3] = FALSE so don't print anything*
- *Count_coordinator ==3 equals Num_Coordinators (3) so drop through to END*

*So the actual output is **Charles***

A pet peeve of mine, which I really find irksome is that the examiners use a left-facing arrow in their pseudocode to represent assignment. It's a royal pain to type into a webpage! The best I can manage is using Wingdings font (a pain) or using a potentially ambiguous substitute such as <= or <-- (which never look right). What's so difficult with using = or even == (two equal signs) to indicate assignment of values? They're commonly used in most languages!

- *Date_required (13/11/2008) == coordinator_booking (13/11/08) so Coordinator_available[count_coordinator, which is 3] == FALSE*
- *Read coordinator_booking (15/11/08)*
- *Date_required (13/11/2008) <> coordinator_booking (15/11/08) so do nothing*
- *Read coordinator_booking (Blank)*
- *Drop through the UNTIL because coordinator_booking == Blank*
- *coordinator_available[3] = FALSE so don't print anything*
- *Count_coordinator ==3 equals Num_Coordinators (3) so drop through to END*

*So the actual output is **Charles***

A pet peeve of mine, which I really find irksome is that the examiners use a left-facing arrow in their pseudocode to represent assignment. It's a royal pain to type into a webpage! The best I can manage is using Wingdings font (a pain) or using a potentially ambiguous substitute such as <= or <-- (which never look right). What's so difficult with using = or even == (two equal signs) to indicate assignment of values? They're commonly used in most languages!

***You** could easily enough understand the meaning of **Count_coordinators = 0** or **Count_coordinators == 0**, couldn't you? And since the syntax of pseudocode is largely arbitrary, who really would care?*

Question 6 - 2008

Question 6

From the variables used in the algorithm, select one of each type to complete the following table.

Type	Variable Name
Boolean Array	Coordinator_Available[]
String (Text) Array	Coordinator_Name[]
Numeric	Count_coordinator or nNum_coordinators

3 marks - average 2.3

This question was generally well answered with most students achieving full marks.

Question A16 - 2009

Question 16

Member ID numbers must be between 1 and fifty thousand (50 000). When a new member is added, the program uses the following code to generate a new member ID number.

Member ID = 1 + int(rand( )* 100000/2)

where rand( ) returns a six-digit random number between zero and one, and

int( ) returns the integer part of whatever number is in the brackets.

If a new member is being added and rand( ) returns 0.002222 then Member ID will be set to

A. 2
B. 12
C. 112
D. 1112

Answer is C. The deskcheck is as follows:

> .002222 * 100,000 = 222.2 (remember your order of operations! * and / have the same priority so work them from left to right)
> 222.2 / 2 = 111.1
> int(111.1) = 111
> 1 + 111 = **112**

It's damned good to see the examiners using some realistic functions in their pseudocode.

I've previously grumbled about restricting pseudocode to the complexity of a four-function calculator. As long as the behaviour of the pseudocode functions is made clear, bring 'em on!

And the question was challenging, but not too hard for the average student.

59% of the state got this right. 19% chose D and the rest were evenly divided between A and B..

Question 7 - 2009

Question 7

During her analysis of the system, Rose interviewed all the assemblers in the Melbourne factory. One assembler mentioned that sometimes RoboCut would reject a good piece of timber.
Rose investigates this and finds that the problem started after the last software upgrade six months ago. She contacts MyCut and it claims that none of its other users have reported this problem. After some argument MyCut agrees to send Rose the algorithms related to the software changes. She finds one algorithm that has to do with the cutting process. RoboCut uses this algorithm to check whether or not a piece of timber is long enough to use.

```
Function Check_Length(Timber_Length, LengthRequired)
Begin
    If Timber_Length > Length_Required Then
        Return true
    Else
        Return false
    End if
End
```

Rose decides to test this algorithm by choosing a length of timber (Timber_Length) of 2.4 metres. For the other variable (Length_Required) she chooses the values 2.3, 2.4 and 2.5.

a. Explain why Rose selected these values.

They are boundary conditions to test the behaviour near, on, and beyond the critical point where the behaviour of the algorithm should change. Most subtle (and hard-to-find) errors tend to occur right at the point where behaviours should change, so test data should focus on that transition point.

3 marks - average 2.0

Following is an example of an acceptable student response.
The values 2.3, 2.4 and 2.5 test all possible number types that could occur. 2.3 is smaller then the required length, 2.4 is equal to the required length and 2.5 is greater then the required length. This comprehensively tests the algorithm.

b. Complete the following table showing what the algorithm should return and what it actually returns.

LengthRequired	What should be returned	What is actually returned
2.3 metres	true	true
2.4 metres	true	false
2.5 metres	false	false

2 marks - average 0.8

c. Explain why RoboCut only sometimes rejects a good piece of timber.

It only accepts pieces **longer** than the length required, and rejects pieces are exactly **equal to** the length required.

1 mark - average 0.6

d. State one alteration to the algorithm that would correct this error.

> should be changed to >=

1 mark - average 0.8

Hmm. A pretty basic > vs >= scenario. Not too imaginative.

• If Timber_Length >= Length_Required Then
• If Timber_Length > Length_Required OR Timber_Length = Length_Required Then
This type of question has traditionally been done poorly by students. This year's algorithm question, though it may have appeared less demanding, still provided many students with a challenge. Most students were able to identify and correct the error; however, many students were still unable to complete the test table correctly or explain why a certain value would be used to test the algorithm. Both of these concepts are highly important when developing software solutions and should be constantly reinforced even when developing minor or small programs throughout the course of the year.

2010 EXAM

The following algorithm applies to Questions 5, 6 and 7.

```
Begin
    Read T
    If T>35 then
        X←'very hot'
    Else
        If T>25 then
            X←'warm'
        Else
            If T>20 then
                X←'perfect'
            Else
                If T<20 then
                    X←'cold'
End
```

Question 6 2010 - Section A

Question 6

When the above algorithm was tested it was found that it did not provide the correct result when T=20. This was caused by a
A. logic error.
B. syntax error.
C. run time error.
D. compile time error.

Answer is A.

A *logic error* is when the code can be interpreted by the compiler, but the algorithm behind the calculation is wrong.
Syntax errors occur when the code cannot be interpreted by the compiler (e.g. an unknown command is used, or the punctuation is not what the compiler expected).
Run time errors occur when a situation arises during program execution that causes a fatal error, such as an unhandled division by zero or running out of memory.
A *compile time* error (I'm guessing) is an obscure error type which would usually be a synonym for syntax error. Maybe an expected library can't be found or something?

72% got this right.

Question 7

Question 7

Test the above algorithm with the value T=25. The variable X will contain
A. 'cold'
B. 'warm'
C. 'perfect'
D. 'very hot'

Answer is C.

T is not > 35 so drop to the first ELSE.

T is not >25 so drop to the next else (line 8).

T is > 20, so 'perfect' is stored in X.

80% got this right.

Question 9 - Section C - 2010

Question 9

During the trial it was found that when nurses changed their passwords for the portable devices the program failed and they were locked out. Investigation found that entering certain characters caused the problem, so it was decided to limit the passwords to just alphabetic letters and numbers. Suzie has suggested the algorithm below to validate a new password before it is stored in the system. She must now test it.

```
Begin
    Get Password
    Charcount←1
    PasswordChar←1st Character of Password
    ValidPassword←True

    Repeat
        PasswordChar←Next Character of Password
        Charcount←Charcount +1
        If (PasswordChar is Not Numeric) Or (PasswordChar is Not Alphabetic) Then
            ValidPassword←False
        EndIf
    Until Charcount=length(password)

    If ValidPassword=False Then
        Print 'Password rejected'
    Else
        Print 'Password accepted'
    EndIf
End
```

Suzie created the test data shown below.

a. For each item of data, give a reason for why that data was chosen.

Test data	Reason for choosing it
12a	It's a valid password and should be accepted by the program
*la	Finds invalid character at the beginning of the password
l*a	Finds invalid characters anywhere in the middle of the password
la*	Finds invalid character at the end of the password

4 marks - average 2.6

To me, it seems pretty simplistic and repetitive for 4 marks' worth. Why not ask students to propose & justify their own test data?

The examiner said: Students struggled significantly with this algorithm question and the related questions. Most students were able to identify that the first value (12a) was valid and the other three items of data were checking an invalid * at the start, middle and end of the password. Students should practise these types of questions either in classroom activities, using past examination papers or in assessment where appropriate.

b. Suggest one other item of data that will test another aspect of the password procedure and explain why it should also be used.

Test data

Null string

Explanation

In case no password is entered, does the procedure behave properly? Especially since it increments the charcount before the password is even looked at!

2 marks - average 0.9

The examiner said: A large number of students correctly suggested that an additional aspect which should be tested was 'blank' or 'null'. However, many students were not awarded any marks as they suggested tests that did not test another aspect of the procedure.

Question 10 - 2010 Section C

Question 10

a. For the algorithm in Question 9, complete the test table below showing what output is expected from the test data and what it actually produces.

Test data	Expected output	Actual output
12a	Password accepted	Password rejected
*la	Password rejected	Password rejected
l*a	Password rejected	Password rejected
la*	Password rejected	Password rejected

2 marks - average 0.7

The examiner said: This question required students to desk check the algorithm and fill in the table. Students needed to fill in each column correctly to be awarded full marks. When testing the algorithm, the content of the table needed to reflect the output as writing in the algorithm, so responses such as 'true', 'false' or 'access granted' were not appropriate.

Suzie found her algorithm does not produce the output expected.

b. Explain why the algorithm is giving the wrong output.

Firstly: In the IF statement, it will reject **all** input because **any** character has to be either not numeric or not alphabetic - or both.

Secondly: even if the IF statement's OR was changed to AND, it would still fail with test data ***la** because the first character of the password is not being checked. Notice how the first character is stored in PasswordChar before the **Repeat**, but the first thing in the **Repeat** construct is loading the *next* character! The first character is never tested.

2 marks - average 0.8

The examiner said: The algorithm had two key errors that students needed to identify:
• the first character in the password is never tested as the first character is always skipped
• the If statement – all possible passwords combinations are rejected. A significant number of students were not able to identify the two key errors. Students need to ensure that they practise these types of questions throughout the year.

c. Explain how the algorithm could be corrected.

Firstly, change the OR in the IF statement to AND.

Secondly, delete lines 3 and 4 of the program.

3 marks - average 0.6

*Are students expected to find **both** problems in the code?*

The examiner said: The algorithm could be corrected in the following ways:

PasswordChar <- Next Character of Password
Charcount <- Charcount +1
Put these lines after the If statement but before the end of the **Repeat Until**

-Or-

Add the **If** statement before the start of the Repeat loop as well as leaving it inside
- **And** -
If (PasswordChar is Not Numeric) AND (PasswordChar is Not Alphabetic) Then
ValidPassword=False
EndIf

The If statement should have the **Or** changed to **And**.