



Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

Examination Answer Books

12 Pages

Instructions to Candidates

1. Please give the information requested in the boxes below.
2. Enter your name in the space provided to the right, fold over and seal down (using the label provided) to hide your name.
3. The answers must be legibly written. The Examiners can either ignore illegible script or have it typed and the expense of typing charged to the candidate.
4. Write your answers in ink.
5. Begin each answer on a fresh page. Do all rough work in the answer book and cross it through.
6. No part of this book is to be torn out; it must be handed in complete.
7. If more books than one are used, fasten them together using the tags available from the invigilator(s).
8. **Every book must be handed in. In no circumstances may answer books, used or unused, be taken out of the examination venue.**

Complete, fold and seal
(see note 2 below)

SURNAME: Friesse

FORENAME(S): Nicholas

1	Student Number	NF173348
2	Title of Examination Paper	Software Development
3	Module Code	ACCA4017
4	Date and Time of Examination	4/1/19 5:10pm
5	Examination Venue	
6	Seat Number	G42
7	Number of Answer Books handed in	1

List the numbers of the questions you have attempted	Office Use
1a	1
1b	1
1c	3
1d	1
2a	10
2b	1+1
2c	10
3a	10
3b	15
4a	3
4b	3
4c	5
Total	64

(Gweler yr ochr arall am y fersiwn Cymraeg)

67/

Question 1

A) A queue data structure can be compared to a ~~line~~ line of people going into a building. It follows the rule of "First in, First Out". So the first person to enter a line (or stored in an array) is the first to leave to make room for others.

This is suitable for handling sets of data at a time and in the order they are stored. Like an appointment system.

B) A Stack data structure can be compared to a stack of plates. As it follows the rule of "Last in, First out". The last item to be added or at the top is removed.

This helps for handling data in any data or storing the most recent values.

C). Recursive means updating itself by using a new value. In the function itself it checks if ~~value~~ value A meets value B. If not, then it calls itself but updating value A.

It can be more efficient than other algorithms as the value being compared can be updated in different ways. More suitable for something like getting a specific value.

D) Object Oriented program is the concept of having something have multiple functions or values. An example would be a car, it can start, move, and stop. But has different components such as the engine, fuel, seats, colour, etc.

But other vehicles like trucks or buses have the same things as a car, but with additional functions and components exclusive to them.

Question 2

B) The reason the result comes out false is due to `dblOutput2` storing a different value. The two numbers ~~used~~ used (`dbl1 - dbl2`) must be done before the `WriteLine`. It must also be converted to `int` as well

Solution:

```
double dblOutput1 = 1.000001;  
double dblOutput2 = 0.000001;
```

```
int dblTotal = Convert.ToInt32(dblOutput1 - dblOutput2);
```

```
Console.WriteLine(dblTotal == 1.0);
```

Question 3

a) Pseudocode:

Function ^{Array} get List of Numbers (~~Array~~ ~~Value~~)

Var Input Num

Print "Number of Values to input: "

Get User Input

is
 If User Input ~~is~~ not ~~input~~ a value:
 Do nothing

Else:

^{Num}
 Input ~~Value~~ = User Input

^{var}
 Loop While ~~is~~ Number less than Input Num:

~~Get~~ User input

Var Value

is

If user input ~~is~~ not ~~input~~ ~~is~~ a value:

Do Nothing

Else:

Value = User Input

~~Store~~ Value in array

Number++

Stop loop IF Number not less than Input Num.

Function CheckForTwos (Array)

Get Array Length

Loop for each Index in Array:

If Index contains ~~2~~ a 2:

Remove value from Array

Stop loop at end of array

Question 3

(continued from a))

Main Program: Start

create ~~array~~ ^{New Array} ~~array~~

Print NewArray

Call Function GetlistofNumbers (NewArray)

Call Function CheckForTwos (NewArray)

Print NewArray

End

Question 4

a)

White box testing - Test the internal components of a program. Looking at the code itself.

Black Box Testing - Test the external components of a program. Looking at what the program does and results produced.

Grey Box Testing - Tests program both ~~internally~~ internally and ~~externally~~ externally. Mix of both Black and White boxes.

b) Unit testing each aspect of a program into separate parts. For example, if a program for manipulating an array (~~inputting~~ inputting new values, displaying values, sorting the array, removing values, etc.) would be tested as separate units.

Testing itself would be done on the methods and functions used for these units, ~~as~~ as well as the values too.

c)

Incremental - Developed and updated into smaller chunks.

Iterative - Developed and updated into repeated development cycles.

Incremental:

Advantage - New features added overtime and easier to get feedback

Disadvantage - Changes are constant or certain features take priority in development.

Question 4

(continued from 4c)

Iterative:

Advantage - Rework a program from scratch. Helps with becoming compatible with modern technology. ✓

Disadvantage - Development focus on New version, Support for older version dropped.