




WEEK 6

 Educators Name: Mr. SC Eilertsen	
Area of Learning: IT	Grade:12
Duration of this unit:	08/6/2020-19/6/2020 – Week 6

Curriculum Outcomes

- Software and Computer Management.
- The Internet.
- Social Implications.
- Algorithm development, pseudocode, trace tables and flowcharts.

What this may look like	Resources/Games/Activities
Theory <ul style="list-style-type: none"> • Internet • Web 1, 2 and 3 • Websites, Blogs and Wikis • Semantic search • SEO • LBS – Location Based Services • Internet Technologies – HTTP, HTTPS, HTML, Javascript, CSS • Internet security services – Firewalls, anti-malware, anti-spam, content filtering, public and private key encryption, digital signatures, SSL • Storing data – online databases • Internet v intranet v extranet • Privacy • Social implications 	Theory <p>Exploring IT. Grade 12. Learning Unit 2. Software and Computer Management. Pages 15 – 22. Check Point 1. Page 22.</p> <p>Exploring IT. Grade 12. Learning Unit 4. The Internet. Pages 28 to 45. Check point 1. Pages 45</p> <p>Exploring IT. Grade 12. Learning Unit 5. Social Implications. Pages 46 – 58. Check point 1 Page 58</p>

<ul style="list-style-type: none"> • Digital divide • E-waste • Environment • Upgrades • Computer crimes, fraud and scams • Internet attacks • Right to access v right to privacy • Social networking • Information overload • Personal information • GUID 	
---	--

Lesson:

- Revision of the points above as necessary.
- Complete the necessary. Check points

Homework Instructions:

- Algorithm development, pseudocode, trace tables and flowcharts based on past examination papers. Question 7. 2018. 2019 and 2019 Supplementary
- Check points as above

QUESTION 7

The library is now going to use an object-oriented programming (OOP) solution rather than a database. **Book** objects will be instantiated and stored in an array of **Book** objects named **bookArr**. Each object will have the following properties that should not be accessible from outside the **Book** class:

bookID – integer
title – string
genre – string
timesBorrowed – integer

7.1 Complete the blank class diagram below to represent the **Book** class. You should use the fields shown above. The **Book** class will need the following:

- A Accessor methods for the **genre** and **timesBorrowed** fields.
- A mutator method for the **title** field. The mutator method should accept a string parameter "t".
- A parameterised constructor method that will accept four parameters, "b", "t", "g", "tb", which correlate to the fields defined above.
- A toString() method that will display all the fields of a **Book** object.

Class name:
Fields:
Methods:

(10)

- 7.2 The librarian would like to know which genre is the most popular. This will be worked out based on the number of times a book is borrowed. Your assistance has been requested to help write an algorithm to answer her question.

7.2.1 What is an algorithm?

(2)

- 7.2.2 Explain why an algorithm can be used to code programs in any programming language.

(2)

- 7.2.3 The algorithm to work out the most popular genre will form part of the **BookArray** class. The **BookArray** class is used to instantiate an array of **Book** objects, and to undertake other tasks. The array named **bookArr** is a field of the class and is declared together with an integer field called **size** to record the number of elements in array.

```
Book [ ] bookArr
```

```
size ← 0
```

A partial algorithm for a typed method called **popularGenre()** is shown below. You are required to complete the algorithm in the space provided. The **bookArr** array begins at index 0.

```
method popularGenre() : String
popular ← 0
position ← 0

//complete the algorithm here

return "The most popular genre is:" +
```

(6)

- 7.3 Another part of the OOP solution has an array that is used to store integer values. There appears to be some duplicate integers in this array, which should not be the case. A programmer has written an algorithm to remove these duplicate values and has coded it, but it is not working correctly. The algorithm he used to code his program has been given to you in **Appendix A**. The line numbers are for reference only. Assume the array has the following values:

intArr[0]	intArr[1]	intArr[2]	intArr[3]	intArr[4]	intArr[5]	intArr[6]
1	2	2	4	6	9	11

To help find the problem, the programmer decided to use a trace table. You need to complete this trace table up to and including the step where size becomes 6 (when the outer loop controlled by variable *i* has executed **once**).

[illegible]

(10)

[30]

57 marks

Total: 180 marks

QUESTION 7

7.1

Book
<ul style="list-style-type: none"> – Integer bookID – String title – String genre – Integer timesBorrowed
<ul style="list-style-type: none"> +Book (Integer b, String t, String g, Integer tb) +getGenre() : String +getTimesBorrowed() : Integer +setTitle(String t) : void +toString() : String

Mark Allocation:

Name of class

All fields private

All fields correctly named and typed

All methods public

Constructor has correct name, correct number of parameters with correct names and types Accept Constructor/Name of class/Create.

All accessor and mutators correctly shown

toString of correct type

Ignore any static methods

If written code: mark just the method headers.

7.2 7.2.1 A series of steps to solve a problem. This is NOT about representing an algorithm – just a definition of what it is.

7.2.2 Because an algorithm is written in pseudocode that can be translated into any programming language for implementation.

7.2.3

```

position ← 0
popular ← 0
size ← number of elements in bookArr
popular ← bookArr[0].getTimesBorrowed()

//action section
loop i ← 1 to size
begin
    if bookArr[i].getTimesBorrowed() > popular
    begin
        popular ← bookArr[i].getTimesBorrowed()
        position ← i
    end if
end loop

return "The most popular genre is:" + bookArr[position].getGenre()
    
```

Accept valid alternatives. There are a number of possible solutions!
Marks allocated accordingly.

Initialise popular

Loop – check merits of loop: from 1 vs from 0

If Provided correct comparison

store of popular

store position

return correct value

Reasonable attempt to assign value to Popular, give a mark for the return even if incorrect.

7.3

Step Number	size	i	k	p	intArr							intArr[i] = intArr[k]?
					[0]	[1]	[2]	[3]	[4]	[5]	[6]	
					1	2	2	4	6	9	11	
1	7											
2		0										
3			1									
4												F
3			2									
4												F
3			3									
4												F
3			4									
4												F
3			5									
4												F
3			6									
4												F
2		1										
3			2									
			✓									
4												T ✓✓
5				2								
6							2					✓
5				3								
6								2				✓
5				4								
6										2		✓
5				5*								
6											2	✓
7	6 ✓											

* ✓✓ for all 4 values correct (2, 3, 4 and 5). If candidate has at least the first value correct (i.e. the '2') allocate one mark.
Check for things happening at the right time!

(10)
[30]

57 marks

Total: 180 Marks

APPENDIX A

intArr[0]	intArr[1]	intArr[2]	intArr[3]	intArr[4]	intArr[5]	intArr[6]
1	2	2	4	6	9	11

NOTE: Array elements are numbered from 0

1	size ← number of elements in intArr
2	for i ← 0 to < size-1
	begin
3	for k ← i+1 to < size
	begin
4	if intArr[i] = intArr[k]
	then begin
5	for p ← k to < size - 1
	begin
6	intArr[p+1] = intArr[p]
	end loop
7	size ← size - 1
	end if
	end loop
	end loop

SECTION E DATA AND INFORMATION MANAGEMENT AND SOLUTION DEVELOPMENT

QUESTION 7

The **MobiHealth** database which stores details of the patients who are treated at each mobile clinic, stores the following data in one table called **tblMedicalData**:

Field	Description
PatientID	Unique ID for each patient treated by the clinic
Surname	The surname of the patient
FirstName	The first names of the patient
ConsultID	Unique ID for each consultation
Date	The date of the consultation
Duration	The duration of the consultation
FollowUp	The number of days when the patient must return for another consultation

A patient can have many different consultations. Each consultation will only be with one patient. Each consultation needs a follow up a certain number of days later.

Sample data for **tblMedicalData** is shown below:

PatientID	Surname	FirstName	ConsultID	Date	Duration	FollowUp
1	Bhebhe	Mthunzi	234	01/04/18	20	10
2	Hlongwa	Kagiso	244	02/04/18	25	90
3	Luthuli	Bonginkosi	267	15/10/18	10	30
4	Chonco	Uluthando	315	12/07/18	15	0
5	Khoza	Langa	337	12/07/18	40	90
6	Zindela	Inyoni	219	21/02/18	55	0
7	Luthuli	Mthunzi	288	16/04/18	12	90
4	Chonco	Uluthando	266	17/06/18	18	10
1	Bhebhe	Mthunzi	322	21/03/18	20	90

7.1 It is common practice to normalise database tables.

7.1.1 List THREE reasons why database tables should be normalised.

Reason 1: _____

Reason 2: _____

Reason 3: _____

(3)

7.1.2 Database tables can be normalised to a number of normal forms.

List the characteristics of both the second and third normal forms. (2NF and 3NF)

2NF: _____

(2)

3NF: _____

(2)

- 7.2 **tblMedicalData** is going to be transformed into TWO new tables: **tblPatients** – for patient specific data and **tblConsultations** – for consultation specific data. In the questions which follow, you must restrict your answers to working with just these two tables, i.e. you may not define any additional tables and you are not required to remove any transitive dependencies.

7.2.1 Define the term *primary key*.

(2)

7.2.2 Define the term *composite key*.

(1)

7.2.3 Write down the best primary key for **tblMedicalData**.

(2)

- 7.2.4 Complete the following two tables, showing the fields that will be present in the two new tables. Your primary keys must be listed as the first field in each table. Indicate your choice of foreign key below the tables.

tblPatients	tblConsultations

Foreign Key: _____ (5)

- 7.2.5 Define the term *foreign key*.

(2)
[19]

QUESTION 8

A software developer is busy writing an OOP application for **MobiHealth**. One of the classes in the OOP application will be used to define **Patient** objects. There are also classes for **Doctor** objects and **Clinic** objects.

Each **Patient** object will have the following fields:

patientID : integer
surname : string
firstName : string
patientAge : integer
medication : array [20] string
clinicName : string
followUp : integer

A patient is always treated by the same doctor and each doctor only works in one mobile clinic, each of which has a unique name.

All fields of **Patient** objects are private.

8.1 Answer the following question relating to the object fields:

In which class(es) will the fields of each **Patient** object be directly accessible? Tick the box next to the correct option below.

- The Patient, Doctor and Clinic classes
- The Patient and Doctor classes
- Only the Patient class
- The Patient and Clinic classes

<input type="checkbox"/>
<input type="checkbox"/>
<input type="checkbox"/>
<input type="checkbox"/>

(1)

8.2 The **Patient** class will have the following methods:

- A parameterised constructor method which will accept 7 parameters named **p** (integer), **s** (string), **fN** (string), **a** (integer), **m** (string array), **c** (string) and **f** (string);
- Accessor method for the **clinicName** field;
- Mutator method for the **clinicName** field. This method will have a parameter: **cn**, of the correct type to match the type of the corresponding field;
- A *toString* method.

8.2.1 The **Doctor** class has a field which is a Patient object. Both classes have *toString* methods. Is this an example of method overloading, method overriding or neither? Explain your answer.

(3)

8.2.2 You are required to assist the developers by completing a class diagram for the **Patient** class.

<u>Fields</u>
<u>Methods</u>

(9)

8.2.3 Currently the age of a patient is stored as an integer field in the **Patient** object. What would be a better data value and corresponding data type to store in this field which will still allow us to know the patient's age?

(1)

- 8.3 When the developers are coding the **Patient** class for the application, they will be working with various algorithms. They first represent the algorithms visually by writing them on pieces of paper which they then fit together to make a working algorithm. Some of the pieces of paper have been mixed up and you have been asked to re-assemble one of the algorithms.

The algorithm in question is meant to sort an array of **medication** (a field in the **Patient** class). The developers are using an **improved Bubble Sort** which uses a **flag** to stop the sort if no further swaps are needed.

- 8.3.1 Included as **Appendix B** is a sheet containing representations of the various pieces of paper that the algorithm was planned on. Each has a letter of the alphabet to identify it. You are required to assemble these into the correct order and write down the correct order, i.e. the letters of the alphabet to get the algorithm to work correctly. NB: Some of the pieces of paper are not relevant to this algorithm. There are exactly the correct number of blocks for a correct answer. You have also been given TWO lines of the sort, G and M. They are in the correct positions.

Sample Answer:

C	G	F	L	D	S	A
---	---	---	---	---	---	---

Your Answer:

				G				M				
--	--	--	--	---	--	--	--	---	--	--	--	--

(10)

- 8.3.2 Does this algorithm sort the array in ascending or descending order?

(1)

- 8.4 A method to determine the average age of a patient is being coded. The algorithm for the method is included as **Appendix C**.

There are four values in the array: 24 ; 36 ; 18 and 22.

The algorithm will loop through an array of patient ages (named **ageArr**) and calculate the average age. It will also maintain a running average whilst looping through the array. If the running average exceeds 60, an error message will be displayed.

- 8.4.1 This algorithm has been coded, but is resulting in unexpected output. You are required to complete the following trace table to assist the programmer in finding the error in the algorithm.

line	size	temp	runningAvg	k	count	runningAvg > 60?	DISPLAY
1	4						
2		0					
3			0				
4					0		
4				0			
6		24					
7					1		
8			24				
9				.		F	
5				1			
6							
7							
8							
9							
5							
6							
7							
8							
9							
10							
5							
6							
7							
8							
9							
10							
11							

(9)

- 8.4.2 Which line in the algorithm is responsible for the error?

_____ (1)

- 8.4.3 Is the variable count necessary in this algorithm as it stands? Can it be replaced by k? Justify your answer.

_____ (2)

[37]

Total: 180 marks
PLEASE TURN OVER

SECTION E DATA AND INFORMATION MANAGEMENT AND SOLUTION DEVELOPMENT

QUESTION 7

7.1 7.1.1 To eliminate repeating groups; to avoid data redundancy; to avoid data anomalies; to make queries simpler.

7.1.2 2NF: Relations are in 1NF and there are no partial dependencies
3NF: Relations are in 2NF and there are no transitive dependencies.
Don't accept relationship diagrams/lists

7.2 7.2.1 A field in a database table which uniquely identifies each record in the table.

7.2.2 A key made up of two or more fields. Accept "two fields"

7.2.3 PatientID ConsultID ConsultID on its own: one mark. PatientID on its own, no mark.

7.2.4

tblPatients	tblConsultations
PatientID	ConsultID
Surname	Date
FirstNames	Duration
ConsultID	FollowUp

OR

tblPatients	tblConsultations
PatientID	ConsultID
Surname	Date
FirstNames	Duration
	FollowUp
	PatientID

Marking: for correct Primary Keys
for fields correct in each table

Foreign Key: ConsultID (Accept PatientID if it is shown in tblConsultations)

7.2.5 A field in a table which is the primary key in another table.

QUESTION 8

8.1 Only the Patient class

8.2 8.2.1 Neither: Because the two methods have the same name but are in unrelated classes. Have to mention "same name" as well as "different classes" for the second and third marks. If candidate gives "Neither" but incorrect explanation, one mark only. If candidate gives "Method overloading" or "Method overriding" (both wrong) but has correct explanation.

8.2.2

PATIENT
Fields - patientID : integer - surname : string - firstName : string - patientAge : integer - medication : array [20] string - clinicName : string - followUp : integer : Accept string (contradiction in question) for all private(-) for all fields and all types correct
Methods + constructor (p:integer, s:string, fN:string, a:integer, m:[] string, c:string, f:string Accept integer (Contradiction in question)) + getClinicName : string + setClinicName (cn : string) + toString : string for all methods public (+) for constructor with all parameters of constructor correct names and types for accessor method for mutator method with correct parameter for correct name and type for toString()

8.2.3 Date of birth will be a better option as you will be able to calculate a patient's age easily. If the age is stored, it will have to change every year.

8.3 8.3.1

K	F	O	R	G	D	C	E	M	B	L	P	U
---	---	---	---	---	---	---	---	---	---	---	---	---

Marking allocation:

K must be before the loop

F to start the outer loop

R must be in the first loop

D, C, E correct sort order

B must be after M
 L to end inner loop
 P to decrement; between inner and outer loops
 U to end outer loop

8.3.2 Descending order

8.4 8.4.1

line	size	temp	runningAvg	k	count	runningAvg > 60?	DISPLAY
1	4						
2		0					
3			0				
4					0		
5				0			
6		24					
6					1		
8			24				
9						F	
5				1			
6		60			2		
7							
8			54				
9						F	
5				2			
6		78					
7					3		
8			80				
9						T	
10							Error
5				3			
6		100					
7					4		
8			105				
9						T	
10							Error
11							25

Marking allocation: for all correct values of temp; allow 1 mark if not all values are correct but candidate's values show a pattern/ follow logically

for all correct values of runningAvg; allow 1 mark if the candidate's values for runningAvg are correct based on incorrect values for temp
 for k 2, 3 AND count 2, 3, 4

for T (True) in last two line 9s and Error in both line 10s; allow 1 mark if candidate's T/T/Error might be wrong but follow his/her solution
 for output value in line 11.

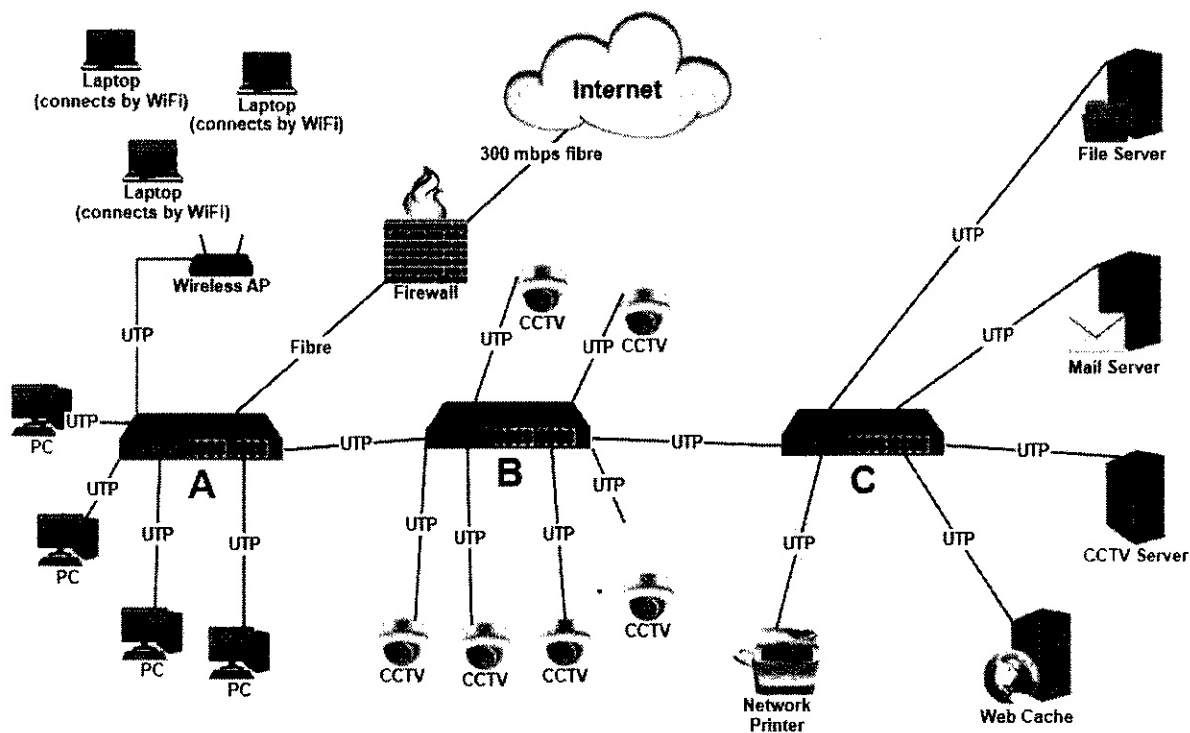
8.4.2 Line 8

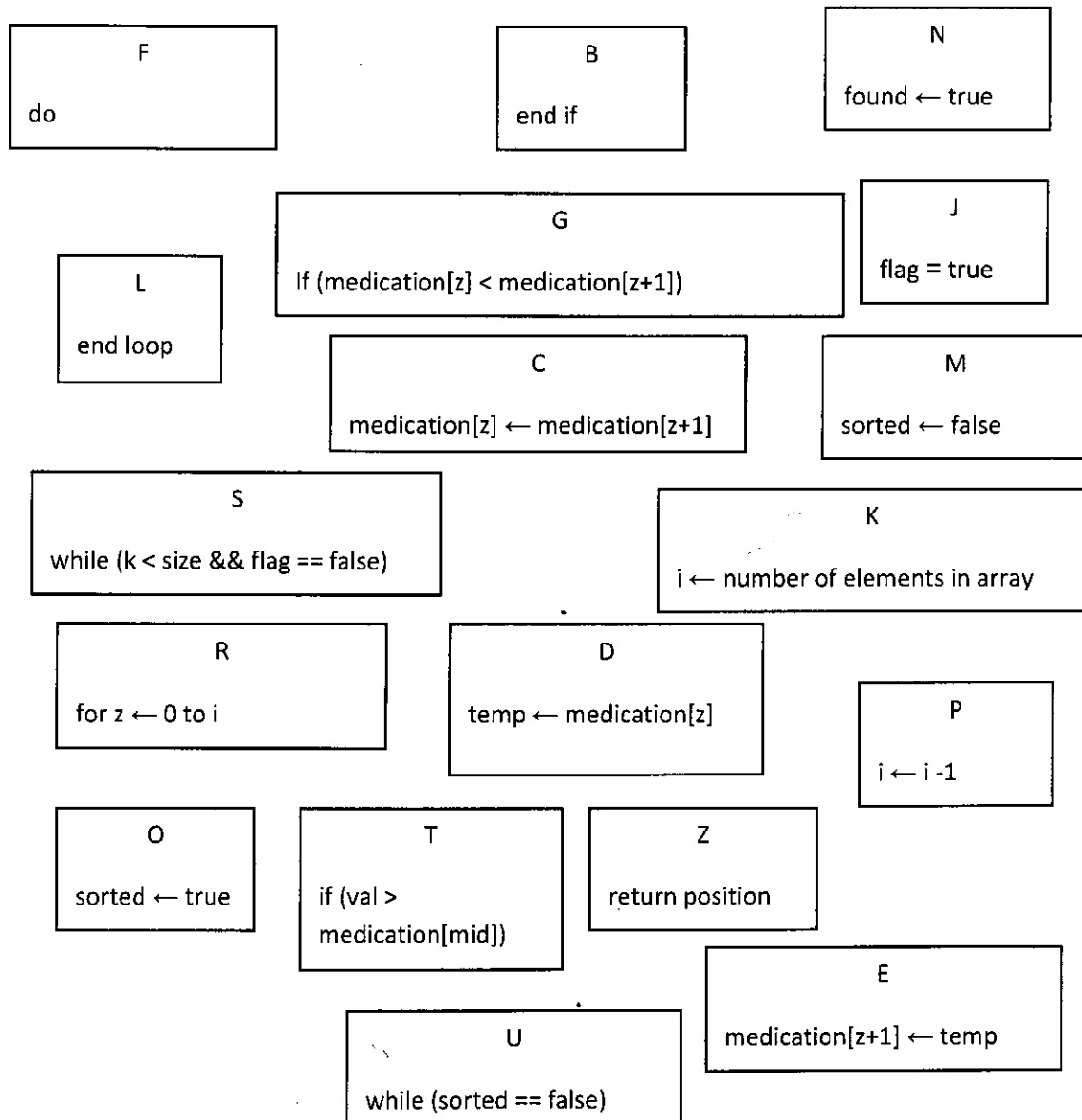
8.4.3 It is necessary; k cannot be used as the algorithm stands as in the first iteration you will have a division by zero.

Total: 180 marks

APPENDIX A

Network Layout



APPENDIX B

APPENDIX C

1	size \leftarrow 4
2	temp \leftarrow 0
3	runningAvg \leftarrow 0
4	count \leftarrow 0
5	for k \leftarrow 0 to size -1
	begin
6	temp \leftarrow temp + ageArr[k]
7	count ++
8	runningAvg \leftarrow runningAvg + (temp / count)
9	if (runningAvg > 60)
10	display "Error"
	end if
	end loop
11	display (temp / count)

