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:

- 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**).

Step Number	size	i	k	p	intArr							intArr[i] = intArr[2]?
					[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												
4												
5												
5												
6												
5												
6												
5												
6												

(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
```