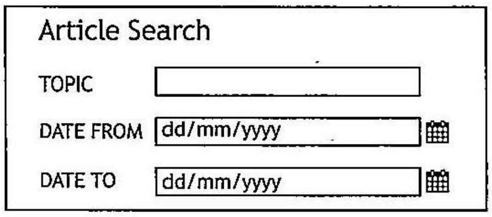
1. (continued)

(b) Two designs for the human computer interface (HCI) of the search facility for the updated website are produced.

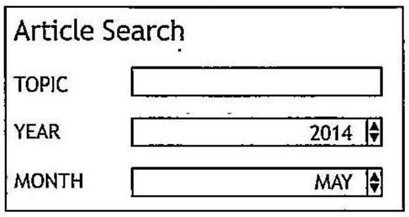
The two designs are shown.

Design 1



Users must type the topic and then either type the date in the required format or select the date from the calendar.

Design 2



Users must type the topic and then select the year and month by using the spinners.

(i) Discuss the suitability of each design for use with a smartphone or tablet device. **2**

The first design is less suitable for a smartphone due to it having a small screen, making it difficult to select a date from the calendar, ~~however~~ The second design would be more efficient as spinners are easy to use when using a small, touch screen device.

(ii) During testing of the search facility, the following list of articles is produced.

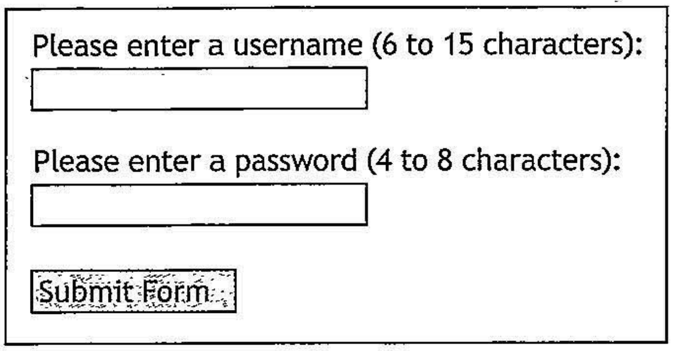
Article Title	*Summary*	*Date*	*Issue*
Processors	Recent processor development	06/05/2016	214
Printers	Inkjet or Laser?	25/03/2016	208
Smartphones	Control your phone by thought	13/05/2016	215

Describe how an insertion sort would reorder the three articles above, listing the articles in chronological order with the most recent article first. **2**

An insertion sort would first compare Processors to Printers, swapping them. It would then insert smartphones and ~~then~~ compare it to Processor, with no swaps taking place, and then compare printers to processors, with no swaps. The order would be Printers then Processors then smartphones.

1. (continued)

(c) An HTML form is used to subscribe to the full service. Part of this form is shown.



Please enter a username (6 to 15 characters):

Please enter a password (4 to 8 characters):

Submit Form

(i) The server-side script called "subscription.php" will receive data from the HTML form.

Write the HTML tags used to generate the subscription form shown above.

~~<form type = "POST", method = subscription.php>~~

```
<form method = "POST" , value = subscription.php >
      Please enter a username
      <input type = text  name = username
         value = " " >


      Please enter a password
      <input type = text  name = password
         value = " " >


      <input type = button  name = submit
         value = " Submit Form " >


<//form>
```

1. (c) (continued)

(ii) Having received the HTML form data, the server-side script "subscription.php" then executes a number of processes. The script

1. assigns the HTML username and password to server-side variables
2. creates a connection with the database server
3. adds data to "member" table of the "subscribedata" database
4. closes the connection

The name of the database server is "magserver" and the username is "subscribe" with the corresponding password "subpass".

Using pseudocode or a server-side scripting language with which you are familiar, write code for processes 1, 2, 3 and 4 described above.

```
<? php

$user = $_POST (username)
$pass = $_POST (pass)

~~$mysql = "database name", "username",~~
~~         "password", "database location"~~

$mysql = "subscribedata", "username",
         "password", "database location"

~~$sqls~~ $sql = "INSERT $user, $pass
           INTO ~~username, password~~
           member.username~~, member.~~
             member.password
           FROM member"

$mysql -> query ($sql)

end $mysql
```

2. Radio Lowden plays songs from the years 1990 to 1999 inclusive. The songs played by the radio station must have featured in the official UK top 40 singles chart from these years.

(a) Using the above example, explain the terms scope and constraints. **2**

The scope is what a program will do - Radio Lowden will play songs from the year 1990 to 1999.
Constraints are what a program will not do - Radio Lowden will not play songs that weren't in the UK top 40

(b) The management of Radio Lowden has commissioned a developer to create a new website for the radio station. One of the pages of the new website will give access to playlists from recent radio programmes.

(i) The developer suggests that the layout and interface of the website belonging to a rival radio station could be copied and used by Radio Lowden.

Discuss whether this is acceptable practice. **2**

This is not acceptable, as the layout and interface of the website are the rival studio's intellectual property. ~~However if they do not~~ If they have a patent for the design copying it is a breach of the copyright, design and patents act. This means it is not acceptable practice. However it would be possible to get away with copying certain aspects of the design, and modifying them, however copying the entire layout and interface is a breach of copyright.

2. (continued)

(c) A PlayList table is used to store details of all playlists created by Radio Lowden and details of each song are stored in a separate table called Song. These tables are part of a relational database.

Sample data for the PlayList and Song tables are shown.

Attribute	Sample
ProgrammeID	1
SongID	A34213
DatePlayed	27/05/15
TimePlayed	09:00

PlayList Table

Attribute	Sample
SongID	A34213
Title	Jack & Dee
Artist	Soozie – L
Year	1997

Song Table

(i) Write the SQL statement which will create the structure of the PlayList table.

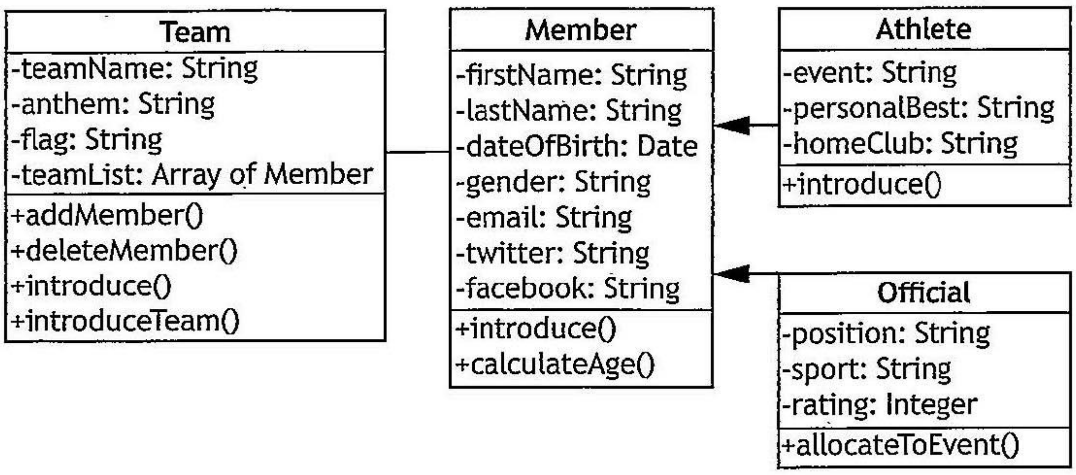
```
~~CREAT~~
~~Ib~~ CREATE TABLE  ~~Blay~~ PlayList

FIELDS  ProgrammeID, SongID,
        Date Played, TimePlayed

UNIQUE KEY  programme ID
```

(ii) Write the SQL query which will list the title of each song played on 26 May 2016.

```
SELECT  Song.Title
FROM    Song, Playlist
WHERE   Song.SongID = PlayList.SongID
        AND  Playlist.DatePlayed=26/05/16
```

3. A program is to be written to process the results of different events in the 2016 Olympic Games.

(a) A simplified version of the UML class diagram for the program is shown.



(i) By referring to the class diagram above, explain:

- the difference between a class and an object
- encapsulation
- inheritance

4

A class is a blue print for an object, for example the Athlete class. An object is instantiated from the class, it is a useable version of the class - for example 'Andy' would be an object of the athlete class.

Encapsulation is when methods and instance variables are hidden to classes outside the one they are in, and can only be accessed through that. ~~Ad~~ addMember() is an example of this, as it changes ~~the~~ teamList variable which can only be accessed through the team class.

Inheritance is where a subclass can access variable, and methods of a superclass. Athlete is a subclass of member, as a result is ~~ba~~ inherits the firstName variable.

3. (a) (continued)

(ii) Some of the code used to define the class Team is provided below.

```
CLASS Team IS         { STRING teamName,
                        STRING anthem,
                        STRING flag,
                        ARRAY OF Member teamList  }

METHODS

    CONSTRUCTOR (  STRING teamName, STRING anthem, STRING flag )
        DECLARE THIS.teamName INITIALLY teamName
        DECLARE THIS.anthem INITIALLY anthem
        DECLARE THIS.flag INITIALLY flag
        DECLARE THIS.teamList INITIALLY []
    END CONSTRUCTOR

    PROCEDURE addMember( Member newMember )
        SET THIS.teamList TO THIS.teamList & [newMember]
    END PROCEDURE

END CLASS
```

An instance of the Team class is to be created using the following values.

Team Name	Brazil
Anthem	Hino Nacional Brasileiro
Flag	Bandeira do Brasil

Using the data provided and a programming language with which you are familiar, write the code used to instantiate a Team object. Your code should make use of each of the values provided. using Java 1

```
Team Brazil = new Team(" Brazil ",
        "Hino Nacional Brasileiro",
        "Bandeira do Brasil" )
```

(b) The details of the athletes taking part in individual events will be stored in separate arrays of objects. For example, the longjumpM array will store the details of all 32 male athletes taking part in the long jump event.

Using a programming language with which you are familiar, write the code used to create the array of objects used to store details of the 32 male athletes in the long jump event. 2

```
Athlete longjumpM = new Athlete[32]
```

3. (continued)

(c) Two introduce methods have been written for the Member and Athlete classes respectively.

```
# Version in Member class
PROCEDURE introduce()
    SEND "Hello, my name is " & THIS.firstName TO DISPLAY
END PROCEDURE

# Version in Athlete class
OVERRIDE PROCEDURE introduce()
    SEND "Hello, my name is " & THIS.firstName TO DISPLAY
    SEND "I'm an athlete on the team" TO DISPLAY
END PROCEDURE
```

A new Team object called myTeam has been created. The following calls have been made to add Ali, Omar and Nour to the team.

```
myTeam.addMember( Athlete("Ali", <only firstName needed here> ) )
myTeam.addMember( Member("Omar", <only firstName needed here> ) )
myTeam.addMember( Official("Nour", <only firstName needed here>) )
```

(i) Write down the output displayed by the following procedure call:

myTeam.introduceTeam **1**

```
Hello, my name is Ali. I'm an athlete on the
team.
Hello, my name is Omar.
Hello, my name is Nour.
```

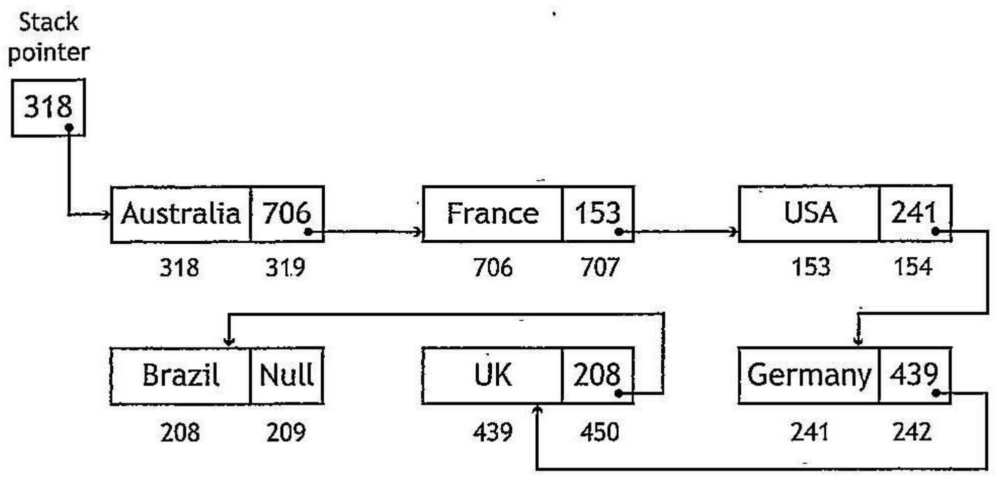
(ii) Use object oriented terminology to explain the operation of the procedure call in (c) part (i) above. **2**

The introduceTeam() method is called for the myTeam object. This will call the introduce() method for Ali, ~~the~~ an object of the Athlete class, Omar, ~~an~~ member of the Member class, and Nour, ~~an~~ member of the member class. The Ali object has a different introduce() method to the member class, due to polymorphism where it can overwrite methods of its super class.

3. (d) (continued)

(ii) The stack storing the medal winning teams could be implemented using a linked list.

The diagram below represents a linked list after the first six teams have been added to the medal table.



Team Russia is to be added to the medal table between Germany and the USA.

Describe how team Russia would be added to the correct place in the linked list.

~~R~~ A variable for Russia would be created. The link between USA and ~~Ger~~ Germany would be broken. USA would then point to Russia linking to it, and Russia would point to Germany.

4. Dawid Mahyne is studying Advanced Higher Computing Science. His teacher has asked him to compare the computational constructs provided by a procedural programming language with those provided by a database.

Dawid starts by creating a database file called "pupils.db". The file contains one table called "pupildata" which stores the pupil data shown.

PupilID	FirstName	LastName	DateOfBirth	RegClass
112211	Joan	Simpson	23/02/1999	6A
112212	John	Adam	12/04/1998	6B
112213	Alison	Brown	30/10/1998	6A
112214	Brian	Morgan	18/11/1998	6C
112215	Bilal	Ali	12/09/1998	6C
112216	Lian	Wong	27/05/1998	6A
112217	Charles	West	23/06/1998	6B
112218	Janet	Smith	18/02/1999	6B
112219	Raymond	Thomas	07/12/1998	6B
112220	Theresa	Cameron	29/01/1999	6A

Dawid writes a program to import the pupil data from the database file and store it in an array of records called "details". His program then applies a binary search to the array of records to display the details of the pupil with PupilID 112213.

(a) (i) Use pseudocode to create the top level design for the program. Your top level design should define the required data structure and call all necessary modules. 3

```
~~1. Import data fro~~
~~1. Create record details~~
1. Create array of records details[10]
     { ~~Stri~~ PupilID : int
       String : FirstName
       String : LastName
       Date : DateOfBirth
       String : RegClass }
2. Write data to details, ~~usin~~
3. display details ~~where pup~~ with
   binary search
```

4. (a) (continued)

(ii) Use pseudocode to refine the binary search used to display the details of the pupil with PupilID 112213. **5**

```
~~loop while unsorted = fa~~
loop until sorted = false
     count = 0
     loop for i = 0 to 8
          if details[i]{pupilID} > details[i+1]{pupilID}
          temp = details[i]
           details[i] = details[i+1]
           details[i+1] = temp
     ~~end loo~~ count +1
           end if
     end loop
     if count = 0
           sorted = true
      end ~~ten~~ if
end loop

~~loop until~~ searchKey = 112213
startpos = 0
end pos = 9
int midPos
loop until found = true OR startPos > endpos
     midPos = (endPos - startPos)/2 rounded down
     if details<pupilID[midPos] < searchKey then
             Start Pos = mid Pos
     else if details{pupilID}[mid Pos] > search Key
     then
             end Pos = mid Pos
     else if details<pupilID>[midPos] = searchKey
     then
          found = true
     endif
end loop
~~Display~~ if found = true then
              display details[midPos]
  end if
```

4. (continued)

(c) Dawid decides to add a new module to his program. This module sorts the data in the array of records into ascending order of registration class. Part of Dawid's code is shown.

```
Line 1    # Name of Sort Algorithm Used: ____________________
Line 2    REPEAT
Line 3        SET swapped TO false
Line 4        FOR counter FROM 1 TO 9
Line 5            IF ______________________________
Line 6                SET swapped TO true
Line 7                < swap data >
Line 8            END IF
Line 9        END FOR
Line 10   UNTIL swapped = false
```

Line 1 and Line 5 of the code are incomplete.

Provide the missing details by rewriting both lines of code. **2**

Line 1 : Bubblesort

Line 5: ~~dat~~ details[count]<details[count+1]

(d) Dawid's school has 2000 pupils.

Explain why it may be more appropriate to use a quick sort rather than the sort algorithm used in part (c) above. **2**

A bubble sort is more efficient for smaller lists, whereas a quicksort is best for long lists. A bubble sort does n(n-1) ~~swaps~~ comparison so ~~[illegible]~~ 3998000 in this case. A quicksort does nlogn so 6602.06 in this case. A quick sort does a lot less comparisons overall thus is more appropriate.

[END OF QUESTION PAPER]