NATIONAL SENIOR CERTIFICATE EXAMINATION
NOVEMBER 2017

# INFORMATION TECHNOLOGY: PAPER II

Time: 3 hours | 120 marks

---

**PLEASE READ THE FOLLOWING INSTRUCTIONS CAREFULLY**

1. This question paper consists of 15 pages. Please check that your question paper is complete.
2. This question paper is to be answered using Object-Oriented Programming principles. Your program must make sensible use of methods and parameters.
3. This paper is divided into two sections. All candidates must answer both sections.
4. This paper is set in programming terms that are not specific to any particular programming language (Java/Delphi) or database (Access/MySQL).
5. Make sure that you answer the questions in the manner described because marks will be awarded for your solution according to the specifications that are given in the question.
6. Only answer what is asked in each question. For example, if the question does not ask for data validation, then no marks are awarded for it, and therefore no code needs to be written.
7. If you cannot get a section of code to work, comment it out so that it will not be executed and so that you can continue with the examination. If possible, try to explain the error to aid the marker.
8. When accessing files from within your code, DO NOT use full path names of the file, as this will create problems when the program is marked on a computer other than the one you are writing on. Merely refer to the files using their names and extensions, where necessary.
9. Your programs must be coded in such a way that they will work with any data and not just the sample data supplied or any data extracts that appear in the question paper. You are advised to look at the supplied data files carefully.
10. Make sure that routines such as searches, sorts and selections for arrays are developed from first principles, and that you do not use the built-in features of a programming language for any of these routines. You may, however, use built-in sub-routines to deep-copy arrays.

11. All data structures must be defined and declared by you, the programmer. You may not use components provided within the interface to store and later retrieve data.

12. Read the whole question paper before you choose a data structure. You may find that there could be an alternative method of representing the data which will be more efficient considering the questions that are asked in the paper.

13. You must save all your work regularly on the disk you have been given, or the disk space allocated to you for this examination.

14. If there is a technical interruption that prevents you from writing your examination, e.g. a power failure, when you resume writing your examinations, you will only be given the time that was remaining when the interruption began. No extra time will be given to catch up work that was not saved.

15. Make sure that your examination number appears as a comment in every program that you code as well as on every page of hardcopy that you hand in.

16. Print a code listing of all the programs/classes that you code. Printing must be done after the examination. You will be given half an hour to print after the examination is finished. Your teacher will tell you what arrangements have been made for the printing of your work.

---

## SCENARIO

InstaPage is a new social networking and instant messaging platform. Social networking platforms allow users to create an online profile and interact with other users' profiles. Instant messaging allows users to exchange text and images on a real time basis. The website and the mobile app for InstaPage will allow users to create a profile, join instant messaging groups, post content, comment on content and respond to content (like, love and dislike). You have been asked to develop a database and a program as prototypes for their final version of the software.

## SECTION A STRUCTURED QUERY LANGUAGE

### QUESTION 1

InstaPage wants to keep track of their users' profiles and the instant messaging groups that they join. They decide to use a three-table database to store this information. A user can create a profile including their username, full name, email address and the privacy of their account (public or private). In addition, the average number of posts per day for each user profile is also recorded. Instant messaging groups are a collection of users that can communicate in real time. When a member of a group sends a message to that group it is received by all members of that group. These messages are not stored in the database. An instant messaging group can have an associated name, creation date, description and group icon. A group must also **have one or more users as administrators** who can add and remove users from the group. The total number of posts that each user has contributed to each of their groups to date has also been recorded.

The database **InstaPageDB** contains three tables. This database will only store the details of user profiles, groups and group memberships, and not the messages themselves. The first table **tblProfiles** contains the details of each user profile that is registered including their username, full name, email address, privacy setting and average number of posts per day. The second table **tblGroups** contains the details of each of the instant messaging groups that have been created including a group name, the date the group was created, a description of the group and the icon file associated with that group. The third table **tblMembership** details the users that are members of groups including whether that user is an administrator of that group and the number of posts that the user has contributed to the group since joining. You are required to extract some useful information from the database.

The fields in the database are discussed below. Below each description is a screenshot of the first 10 rows of data for your convenience. The tables do contain more data:

**tblProfiles**

| Field Name | Data Type | Description |
|---|---|---|
| ProfileID | Number | This field assigns a unique ID for each item as a number. |
| Username | Text | This field contains the username associated with this profile as text. |
| Fullname | Text | This field contains the full name of the user as text. |
| Email | Text | This field contains the email address of the user as text. |
| Privacy | Text | This field contains the privacy setting of a user's profile. Possible values are "public" and "private". |
| DailyPostRate | Number | This field contains the average number of posts the user makes per day to their groups as a real number. |

| tblProfiles | | | | | |
|---|---|---|---|---|---|
| **ProfileID** | **Username** | **Fullname** | **Email** | **Privacy** | **DailyPostRate** |
| 1 | kmoore0 | Kenneth Moore | kmoore0@mashable.com | public | 24.2 |
| 2 | sberry1 | Stephanie Berry | sberry1@unesco.org | public | 11.9 |
| 3 | ahernandez2 | Ashley Hernandez | ahernandez2@psu.edu | private | 6.5 |
| 4 | lruiz3 | Lillian Ruiz | lruiz3@nhs.uk | private | 13.9 |
| 5 | jlarson4 | Jimmy Larson | jlarson4@cafepress.com | public | 18.4 |
| 6 | showard5 | Sara Howard | showard5@surveymonkey.com | private | 19.2 |
| 7 | jtaylor6 | James Taylor | jtaylor6@mlb.com | public | 14.2 |
| 8 | cphillips7 | Charles Phillips | cphillips7@google.pl | public | 11.1 |
| 9 | jfreeman8 | Jeremy Freeman | jfreeman8@ovh.net | private | 1.6 |
| 10 | sortiz9 | Shawn Ortiz | sortiz9@csmonitor.com | public | 10.1 |

## tblGroups

| Field Name | Data Type | Description |
|---|---|---|
| GroupID | Number | This field assigns a unique ID for each group. |
| GroupName | Text | This field contains the name of the group as text. |
| CreatedDate | Date/Time | This field contains the date that the group was created as a date/time. |
| Icon | Text | This field contains the file name of the group icon as text. Possible file extensions are "png", "jpeg", "jpg" and "gif". |
| Description | Text | This field contains a description of the group as text. |

| tblGroups | | | | |
|---|---|---|---|---|
| **GroupID** | **GroupName** | **CreatedDate** | **Icon** | **Description** |
| 1 | Cool Kids | 2016-10-19 | consectetuer_eget.gif | No nerds allowed |
| 2 | The Secret Tribe | 2017-05-11 | totem.jpeg | The illuminati are coming |
| 3 | Tech Gurus | 2016-10-29 | ssd.png | Ask us anything |
| 4 | Ziemann and Sons | 2017-03-21 | family.png | The family men |
| 5 | The Pi Eaters 314 | 2016-10-08 | golden_ratio.jpeg | All your math questions answers |
| 6 | Smith Family | 2017-07-05 | granny.gif | Smith Family Group |
| 7 | The Band of Brothers | 2017-02-06 | insignia.png | Military chat |
| 8 | Girls just wanna have fun | 2016-10-31 | saturday.jpeg | Party hardy |
| 9 | Rugby Group | 2016-03-02 | ball_skills.png | Touch pause engage |
| 10 | Toys for Boys | 2016-03-22 | ps4.png | Gadgets and stuff |

**tblMembership**

| Field Name | Data Type | Description |
|---|---|---|
| GroupID | Number | This field contains the ID of the group that a user belongs to. This field is a foreign key to tblGroups. |
| ProfileID | Number | This field contains the ID of the profile/user that belongs to the group. This field is a foreign key to tblProfiles. |
| Admin | Boolean | This field contains whether the user is an administrator of that group as a boolean. |
| Posts | Number | This field contains the number of posts that each user has contributed to that group as a number. |

| tblMembership | | | |
|---|---|---|---|
| **GroupID** | **ProfileID** | **Admin** | **Posts** |
| 1 | 8 | False | 429 |
| 1 | 18 | False | 276 |
| 1 | 28 | False | 89 |
| 1 | 29 | False | 107 |
| 1 | 35 | False | 118 |
| 1 | 45 | True | 497 |
| 1 | 47 | False | 897 |
| 1 | 54 | False | 320 |
| 1 | 57 | False | 123 |
| 1 | 63 | True | 246 |

1.1 Write a query that will display all user details from **tblProfiles**. Sort the results in descending order of **DailyPostRate**. (3)

1.2 Write a query that will display the **Username** and **Fullname** of all profiles from **tblProfiles** that have their **Privacy** set to "public". (3)

1.3 Write a query that will add the Raymond Ellis (**ProfileID** 98) to the "Girls just wanna have fun" group (**GroupID** 8) as an administrator. He is new to the group so his posts for that group can be set to zero. (4)

1.4 Write a query which will display the average number of posts that each group's members post along with that group's **GroupID**. Round off the average number of posts to 1 decimal place. A sample of the first 10 rows of correct output are given below:

| GroupID | AvgPosts |
|---|---|
| 1 | 321.3 |
| 2 | 529.2 |
| 3 | 425.7 |
| 4 | 523.6 |
| 5 | 451.7 |
| 6 | 578.1 |
| 7 | 472.5 |
| 8 | 490.9 |
| 9 | 520.2 |
| 10 | 529.5 |

**Note:** You do not need to display the group name, only the groupID. (4)

1.5 Write a query that will display the months of the year in which more than 2 groups were created. You should display the month number in a field called **CreatedMonth** and the total number of instant messaging groups created in each of those months as a field called **TotalGroups**. The correct output is given below:

| CreatedMonth | TotalGroups |
|---|---|
| 3 | 3 |
| 10 | 4 |

(5)

1.6 Write a query that will determine the total number of messages an instant messaging group accumulates in a week. In order to calculate this you must use each members' average posts per day (**stored in tblProfiles**). You can assume that all of a user's posts will contribute to each of the groups to which they belong. A sample of the first 10 rows of correct output are given below:

| GroupID | PredictedWeeklyPosts |
|---|---|
| 1 | 1138.9 |
| 2 | 1224.3 |
| 3 | 1440.6 |
| 4 | 1565.9 |
| 5 | 1782.2 |
| 6 | 1316.7 |
| 7 | 1293.6 |
| 8 | 560 |
| 9 | 898.1 |
| 10 | 1312.5 |

(6)

1.7 Write a query that will display how many administrators in each group have private profiles. You must display the name of each group and the number of administrators with private profiles. Keep in mind that an instant messaging group can have more than one administrator. The correct output is given below:

| GroupName | PrivateAdministrators |
|---|---|
| Cool Kids | 1 |
| Dank Memes | 2 |
| Gossip Groupies | 1 |
| Movie Chat | 1 |
| Rugby Group | 1 |
| Smith Family | 2 |
| Tech Gurus | 2 |
| The Pi Eaters 314 | 1 |
| The Untouchables | 1 |
| Toys for Boys | 1 |
| Ziemann and Sons | 1 |

(9)

1.8 Write a query which will change the icon extensions of all group icons that have the extension "jpeg" to "jpg". For example if the icon is named "myfile.jpeg" you must change it to "myfile.jpg". (6)

**40 marks**

## SECTION B OBJECT-ORIENTED PROGRAMMING

InstaPage allows users to post content to their social networking platform. A user can post content (text, URLs, images, videos) to the site. Other users can then respond to a user's post by commenting, liking, disliking or loving their post. A user can post multiple times. A user can comment on a post as well as reacting to a post (liking, disliking or loving). Multiple users can react to or comment on a single post. If a user comments on a post and responds to the same post, this is recorded as two different events.

Each **Post** has the following information:
- id – the unique id of the post
- user – the user that authored the post
- postContent – the content of the post
- postDateTime – the date and time the post was originally posted in the format YYYY-MM-DD HH:mm

Responses are associated with a particular post. Each **Response** contains the following information:
- postID – the ID of the post that the reaction is associated with
- user – the user that reacted to the post
- comment – the comment that the user added to the post (note that this will be blank for reactions such as likes, dislikes and loves)
- responseDateTime – the date and time of the response in the format YYYY-MM-DD HH:mm
- responseType – the type of reaction to the post. Possible values are like, comment, dislike or love

It is important to note that a response that is a comment will have the responseType "comment" and the comment will be stored in the field called **comment**. If the **responseType** to a post is a reaction (such as "like", "dislike" or "love") the comment field will be left blank.

Response types are stored as integers as follows:
- like = 1
- comment = 2
- dislike = 3
- love = 4

You have been given a file called **data.txt** that contains the information of an unknown number of posts and responses to those posts. The file is in no particular order. A sample of the first 10 lines of the file is given below:

```
1#d0ugSm1th#Merry Christmas everyone!#2016-12-25 10:45
1#susanSmith89#To you too Doug!#2016-12-25 10:45#2
1#susanSmith89##2016-12-25 11:02#4
1#BobSeegers##2016-12-25 13:21#1
2#s1ckboy77#Loving life!#2017-09-25 16:41
2#LizelleDeBruin#You have plenty to be grateful for.#2017-09-25 16:41#2
2#Kwezi88#Love your positive attitude.#2017-09-25 16:44#2
2#Kwezi88##2017-09-25 17:49#4
2#LeRouxLight##2017-09-25 17:49#1
3#L3xiJones#Struggling to sleep at night. Any advice?#2017-06-25 00:08
```

Each line of this file contains the information of a single post or a single response. Lines containing posts are formatted as follows:

```
id#user#postContent#postDateTime
```

Lines containing responses are formatted as follows:

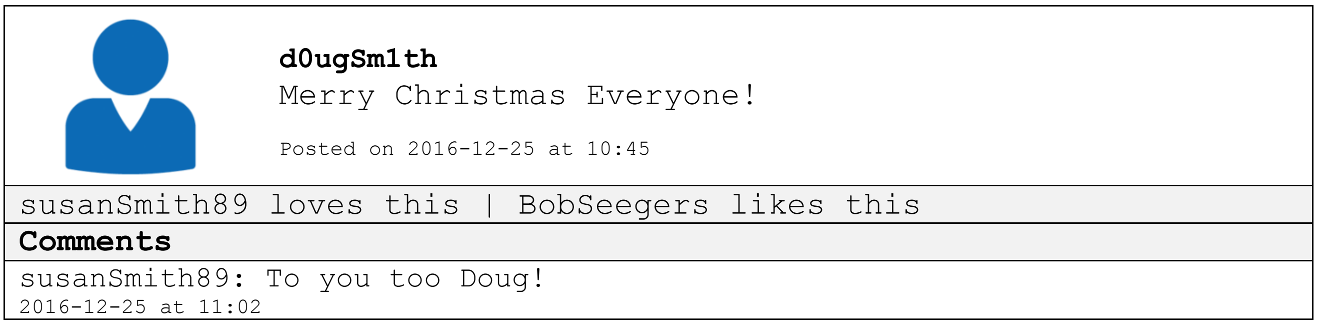
```
postID#user#comment#responseDateTime#responseType
```

The postID will refer directly to a post entry in the same file. E.g. postID 1 on lines 2, 3 and 4 refer to the post with ID 1 on the first line of the file.

The first line of the file data.txt is the original post by **d0ugSm1th** where he wishes everyone for a merry Christmas at 10:45 on 25 December 2016.

Lines 2, 3 and 4 are all responses to his post. Notice that line 2 is a comment response (responseType is 2) by **susanSmith89** and thus has the comment portion present. However, line three of the file is a response "love" (responseType is 4) by the same user, **susanSmith89**, to the post with ID 1. This means no comment is present for the second response. The third response (on line 4) is another user, **BobSeegers** who liked the post.

The image below is a graphical representation of the first post and its three responses:



**d0ugSm1th**
Merry Christmas Everyone!
Posted on 2016-12-25 at 10:45

susanSmith89 loves this | BobSeegers likes this

**Comments**

susanSmith89: To you too Doug!
2016-12-25 at 11:02

## QUESTION 2

Use the class diagram below to create a new class called **Post**. This class will be used to create objects that will store the details of one post. The diagram below indicates the properties and methods that are required. Take careful note of the method and parameter names in the UML diagram.

| **Post** |
|---|
| **Properties:**<br>– integer id<br>– String user<br>– String postContent<br>– String postDateTime |
| **Methods:**<br>+ Constructor(integer inID, String inPostUser, String inPostContent, String inPostDateTime)<br>+ getPostID : integer;<br>+ toString() : String |

2.1 Write code to create a new class called **Post**. (1)

2.2 Write code to create the four properties for the **Post** class as indicated in the above class diagram. (3)

2.3 Write code to create a constructor method that accepts one integer and three Strings as parameters which represent the id of the post, the post user, post content and the date and time of the post respectively. These parameters should be used to set the values of the four properties of the **Post** class. (3)

2.4 Write code to create an accessor method for **id** property of the **Post** class. This method should be named appropriately and return the correct type and value for the **id** property. (1)

2.5 Write code to create a **toString** method that will return a string representing the post's information. The format is as follows:

**postDateTime**<space>**user**<space>"posted:"<space>**postContent**

for example:

```
2016-12-25 10:45 d0ugSm1th posted: Merry Christmas everyone!
```
(2)

**[10]**

## QUESTION 3

Use the class diagram below to create a new class called **Response**. This class will be used to store the details of one response to a particular **Post**. Note that there are class constants defined in the class diagram. The diagram below indicates the properties and methods that are required.

**Response**

**Properties:**

– integer postID
– String user
– String comment
– String responseDateTime
– integer responseType

+ RESPONSE_LIKE : integer = 1
+ RESPONSE_COMMENT: integer = 2
+ RESPONSE_DISLIKE: integer = 3
+ RESPONSE_LOVE: integer = 4

**Methods:**

+ Constructor(integer inPostID, String inUser, String inComment, String inResponseDateTime, integer inResponseType)
– getReactionType() : String
+ getPostID : integer
+ toString() : String

3.1 Write code to create a new class called **Response**. (1)

3.2 Write code to create five properties that will store the **postID, user, comment, responseDateTime** and **responseType**. Chose the appropriate data types for these attributes. These properties should not be visible from outside the class. (3)

3.3 Write code to create four class constants as integers that will represent the different types of reactions. The values for each **responseType** are given in the class diagram. (3)

3.4 Write code to create a constructor method that will initialise all the properties of the **Response** class. Note the **inResponseType** parameter is an integer. Use the parameters to initialise the properties of the **Response** class. (3)

3.5 Write code to create a method called **getResponseType** which returns a string. This method should test the integer value of the **responseType** attribute **using the class constants** and return a string according to the value.

If **responseType** has a value of RESPONSE_LIKE, return "liked" as a string.
If **responseType** has a value of RESPONSE_COMMENT, return "commented" as a string.
If **responseType** has a value of RESPONSE_DISLIKE, return "disliked" as a string.
If **responseType** has a value of RESPONSE_LOVE, return "loved" as a string. (3)

3.6 Write code to create an accessor method for **postID** property of the **Response** class. This method should be named appropriately and return the correct type and value for the postID property. (1)

3.7 Write code to create a **toString** method that will return a string comprised of the information for the response. The string returned should be in the following format:

**responseDateTime**<tab>**user**<space>**responseType**<space>"this post"

For example:

```
2016-12-25 13:21  BobSeegers likes this post
```

The output for comments is different as it should display the comment.

**responseDateTime**<tab>**user**<space>**responseType**<space>"on this post:" <space>**comment**

For example

```
2016-12-25 11:02  susanSmith89 commented on this post: To you too
Doug!
```

(3)
**[17]**

## QUESTION 4

4.1 Write code to create a new class called **InstaPageManager**. (1)

4.2 Write code to declare the following TWO arrays and TWO counter variables as instance variables:

- An array that can be used to store up to 100 **Post** objects.
- An array that can be used to store up to 500 **Response** objects.
- A counter variable to keep track of the number of **Posts** objects stored.
- A counter variable to keep track of the number of **Response** objects stored. (5)

4.3 Write code to create a constructor method that will read the contents of a file of posts and reactions. The file name can be hardcoded in the constructor method. Each line will result in ONE **Post** object **OR** ONE **Response** object being added to the respective array. Do the following:

- Check if the file exists. If it does not, display an error message.
- Open the file for reading.
- Loop through the file until there are no more lines. In each iteration of the loop:
  - Read in each line and split the **Post** or **Response** data into separate items.
  - Create a **Post** or a **Response** object depending on the number of data items on that line.
  - Add the **Post** or **Response** object to the appropriate array. (13)

4.4 Write code that will create a method called **getAllPosts**. This method should return a string that contains the information of all **Post** objects in the array that stores these objects. Use the object's **toString** methods that you created in the questions above. Each posts' details should be on a new line. (5)

**[24]**

## QUESTION 5

5.1 Write code to create a simple user interface called **InstaPageUI** that will allow simple output. (1)

5.2 Declare and instantiate an **InstaPageManager** object at the appropriate place in the code. (1)

5.3 Write code that will display all the **Post** objects by calling the appropriate method in the **InstaPageManager** class. (1)

**[3]**

## QUESTION 6

You are now required to write code to create a new class called **PostWithResponses** that will store the details of a **Post** object as well as an array of associated **Response** objects.

6.1 Declare a new class called **PostWithResponses**. (1)

6.2 Your class should be capable of storing all the details of a **Post** object as well as an array of associated **Response** objects. Declare attributes that will store this data. Each **PostWithResponses** object will only store the details of a single post and an array of **Response** objects associated with that **Post**. You may reuse existing classes instead of redeclaring attributes. (3)

6.3 Write code to create a constructor method for this class that will initialise both attributes of the class. (4)

6.4 Write code to create **toString** method that will return a string with the information of the **Post** and its associated reactions in the following format:

```
<PostDetails><newline><tab>"Responses"
<newline><tab><tab><ResponseDetails1>
<newline><tab><tab><ResponseDetails2>
<newline><tab><tab><ResponseDetailsN>
```

For example:

```
2016-12-25 10:45 d0ugSm1th posted: Merry Christmas everyone!
   Reactions:
   2016-12-25 10:45 susanSmith89 commented on this post: To you too
   Doug!
   2016-12-25 11:02 susanSmith89 loved this post
   2016-12-25 13:21 BobSeegers likes this post
```

**Note:** The text/line wrapping of your program's output may differ. (5)

**[13]**

## QUESTION 7

Add code to the **InstaPageManager** class that will populate an array of **PostWithResponses** objects and display the contents of the array.

7.1 Add code to the **InstaPageManager** class that will populate an array of **PostWithResponses** objects. Each object in the array should store the details of one **Post** object in the original array and an array of associated **Response** objects.

Use the **getID** method of the **Post** class and the **getPostID** method of the **Response** class to ascertain which reactions belong to which post.

This code should return a string consisting of all output of the **toString** method of each **PostWithResponses**. (12)

7.2 In the **InstaPageUI** call the method you created in 7.1 and display the results.

The correct output is given below:

```
2016-12-25 10:45 d0ugSm1th posted: Merry Christmas everyone!
   Reactions:
   2016-12-25 10:45 susanSmith89 commented on this post: To you too
Doug!
   2016-12-25 11:02 susanSmith89 loved this post
   2016-12-25 13:21 BobSeegers likes this post

2017-09-25 16:41 s1ckboy77 posted: Loving life!
   Reactions:
   2017-09-25 16:41 LizelleDeBruin commented on this post: You have
plenty to be grateful for.
   2017-09-25 16:44 Kwezi88 commented on this post: Love your
positive attitude.
   2017-09-25 17:49 Kwezi88 loved this post
   2017-09-25 17:49 LeRouxLight likes this post

2017-06-25 00:08 L3xiJones posted: Struggling to sleep at night.
Any advice?
   Reactions:
   2017-06-25 00:08 ReggieSanders53 loved this post
   2017-06-25 08:45 TheSciGuy commented on this post: Try some
rooibos tea before bed.
   2017-06-25 09:54 Ins0mnia commented on this post: If you find a
solution please let me know.
   2017-06-26 03:22 SiyaSandton dislikes this post

2017-01-25 21:45 GuruZA posted: Trump is such a terrible president!
   Reactions:
   2017-01-25 21:45 JumpingJill2 commented on this post: That's
democracy for you.
   2017-01-25 22:45 JumpingJill2 likes this post
   2017-01-25 22:45 ReggieSanders53 commented on this post: Big
business really likes him.
   2017-01-27 21:33 PerfectPete likes this post
   2017-01-27 23:54 Fikile66 dislikes this post
```

**Note:** The text/line wrapping of your program's output may differ. (1)

**[13]**

**80 marks**

**Total: 120 marks**