



eduresourceza.com

PRELIM 2019
Information Technology
Paper 2 Practical

TIME: 3 HOURS

MARKS: 120

EXAMINER: CLUSTER

MODERATOR: CLUSTER

This question paper consists of 10 pages (cover page included)

SURNAME: «Surname»

USER NAME: «sAMAccountName»

PASSWORD: «Password»

PLEASE READ THE FOLLOWING INSTRUCTIONS CAREFULLY

1. Log into the exam profile allocated to you.
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 (Delphi/Java) 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. 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.
9. Make sure that routines such as searches, sorts and selections are to be developed from first principles, and that you do not use the built-in features of a programming language for any of these routines.
10. 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.
11. 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.
12. You must save all your work regularly on the disk you have been given, or the disk space allocated to you for this examination.
13. 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.
14. Make sure that your name appears as a comment in every unit that you code as well as on every page of hard copy that you hand in.
15. 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

Modern day movies may use live action, CGI (computer-generated imagery) or some combination of both. Between actor salaries and years of special effects editing the cost to produce a movie can run into the hundreds of million US dollars. Studios need to carefully track movie budgets, advertising budgets, audience numbers and merchandising income in order to ensure they make a profit on the movie.

SECTION A**SQL****[40]****Question 1**

The given database called **Movies** contains three tables: **tblMovie**, **tblDirector** and **tblMovieDirectors**. The data from the tables are detailed below with screenshots of a selection of the data contained in each table.

Design of tblMovies

tblMovie X			
	Field Name	Data Type	Description (Optional)
	MovieID	AutoNumber	This assigns a unique ID to each movie
	MovieName	Short Text	This field contains the name of the movie
	ReleaseDate	Date/Time	This field contains the release date of the movie
	Budget	Number	Total budget of the movie in \$US
	RunningTime	Number	This field contains the running time of the movie in minutes
	Language	Short Text	This field contains the original language of the movie
	BoxOffice	Number	Total box office takings in \$US

Sample of tblMovie

tblMovie X							
	MovieID	MovieName	ReleaseDate	Budget	RunningTime	Language	BoxOffice
+	1	Avengers: Endgame	2019/04/22	356	181	English	2743
+	2	Avengers: Infinity War	2018/04/23	316	149	English	2048
+	3	Avengers: Age of Ultron	2015/04/13	444	141	English	1405
+	4	The Avengers	2012/04/11	220	143	English	1519
+	5	Much Ado About Nothing	2013/06/21	2.4	108	Early Modern English	5.3
+	6	The Big Lebowski	1998/01/18	15	117	English	46.2
+	7	No Country for Old Men	2007/05/19	25	122	English	171.6
+	8	The Intouchables	2011/09/23	10.8	112	French	360
+	9	Those Happy Days	2006/06/28	7.2	103	French	10.9
+	10	47 Ronin	2013/12/25	175	127	English	151.8

Design of tblDirector

tblDirector X		
Field Name	Data Type	Description (Optional)
DirectorID	AutoNumber	This field contains the ID of the director
FirstName	Short Text	This field contains the first name of the director
Surname	Short Text	This field contains the surname of the director
DateOfBirth	Date/Time	This field contains the date of birth of the director

Sample of tblDirector

tblDirector X				
	DirectorID	FirstName	Surname	DateOfBirth
+	1	Anthony	Russo	1970/02/16
+	2	Joseph	Russo	1971/06/28
+	3	Joseph	Montgomery	1994/08/13
+	4	Joel	Coen	1954/11/29
+	5	Ethan	Coen	1957/09/21
+	6	Steven	Myers	1973/04/17
+	7	Eric	Toledano	1971/06/03
+	8	Carl	Rinsch	1982/04/12
+	9	Rob	Cohen	1949/03/12
+	10	Ivan	Reitman	1946/10/27

Design of tblMovieDirectors

tblMovieDirectors X		
Field Name	Data Type	Description (Optional)
MovieID	Number	This field contains the ID of the movie
DirectorID	Number	This field contains the ID of the director

Sample of tblMovieDirectors

tblMovieDirectors X		
MovieID	DirectorID	
1	1	
1	2	
2	1	
2	2	
3	14	
4	14	
5	14	
6	4	
6	5	
7	4	

Write SQL statements to help a producer, Gigi Pritzker, with the following queries:

- 1.1 She needs a list of movies displaying all the details of every movie sorted by release date from newest release to the oldest release and then by the movie name alphabetically. (3)
- 1.2 Display the movie which had the highest budget. (2)
- 1.3 She discovers there was a mistake in the calculations of box office takings for every movie. Update the **BoxOffice** column by multiplying every value by 0.95. (3)
- 1.4 For her next movie she needs to select a director who is older than 50 in order to improve her age equity rating. Display the **FirstName** and **Surname** of directors who are older than 50 as of the end of the year, also display their ages. (4)
- 1.5 Steven Spielberg will not be directing any movies produced by Gigi Pritzker. Remove him from the table of directors. (3)
- 1.6 Each director needs a username to login to the database. Generate a username for each director by joining their surname and the first letter of their first name. For example, Joseph Russo's username would be RussoJ. Display the **DirectorID** and their username. Sample output is below. (4)

DirectorID	Expr1001
1	RussoA
2	RussoJ
3	MontgomeryJ
4	CoenJ
5	CoenE
6	MyersS
7	ToledanoE
8	RinschC
9	CohenR
10	ReitmanI

- 1.7 A sequel to the Avengers is planned. It will be called "Avengers:Still Not Dead". The release date will be 2021/09/01, the budget is 360, running time is 167, language is English. Put in an estimate for the box office takings which is the average of every movie which has a title STARTING with Avengers. (5)
- 1.8 Some movies have not yet been linked to directors. Display a list of the **MovieIDs** and **MovieNames** which have no director information yet. (4)

- 1.9 A new foreign film award for directors is announced. All directors who have directed any movie NOT filmed in any variety English are eligible. Display the **FirstName** and **Surname** of all directors who are eligible. The director's first name and surname should only be displayed once. Sample output is below. (6)

FirstName	Surname
Advait	Chandag
Eric	Toledano
Olivier	Nakache
Rajkumar	Hirani

- 1.10 Gigi Pritzker wants to see how much profit each director in the database has made. Display a list with the **DirectorID**, number of movies made (**NumMovies**) and total profit (**Profit**) (box office takings – budget). An example of the output is below: (6)

DirectorID	NumMovies	Profit
1	2	4119
2	2	4119
4	2	177.8
5	2	177.8
6	1	349.2
7	2	352.9
8	1	-23.2
9	3	147
10	2	25.3

SECTION B STARTS ON THE NEXT PAGE

Avengers: Endgame was a massive success and it has been decided to create an Avengers video game to tie in with the movie. The game will be 2D (two dimensions, across the screen and down the screen). The tutorial level of the game will feature superheroes and training dummies. Each superhero or training dummy will be located at a position on the screen indicated by a positive x (number of pixels from the left of the screen) and y value (number of pixels from the bottom of the screen). Superheroes will be able to attack the training dummies (which can't fight back). You will be required to implement code to determine whether a superhero is within range of a training dummy.

A position is represented by the following:

- **xPos** – the x co-ordinate
- **yPos** – the y co-ordinate

Each training dummy has the following details:

- **name** – the descriptive name of the character
- **posi** – the combined **xPos**, and **yPos** of the object
- **health** – an integer measuring how much health the training dummy has

Each superhero has the following details:

- **name** – the descriptive name of the character
- **posi** – the combined **xPos**, and **yPos** of the object
- **health** – an integer measuring how much health the training dummy has
- **damage** – an integer measuring how much damage the superhero can do with an attack.
- **range** – an integer number indicating the maximum distance the target of an attack can be, measure from the superhero's **posi**.

You have been given a file called **fighters.txt** that contains the information of an unknown number of superheroes and training dummies.

For example "Okoye#33#77#100#32#14" is the data for a superhero called "Okoye" in position (33,77) with health of 100, damage of 32 and range of 14. The training dummy indicated by "Training Dummy 11#94#57#93" has the name "Training Dummy 11" in position (94,57) with a health of 93.

Following is the first 10 lines of data from the text file:

```
Okoye#33#77#100#32#14
Avenger Captain America#69#17#59#17#12
Training Dummy 11#94#57#93
Ant-Man#3#84#156#10#17
Falcom#65#82#104#37#17
Wasp#28#77#68#24#11
Training Dummy 1#80#29#30
Spider-Man#85#73#162#34#19
Doctor Strange#69#33#152#27#10
Rocket#76#4#77#28#17
```

Each line of this file contains the information of a superhero or training dummy.

QUESTION 2

9 MARKS

Use the class diagram below to create a new class called **Position**. This class will be used to create objects that will store the position of a training dummy or superhero on the screen. The diagram below indicates the fields and methods that are required.

Position
Properties: - xPos : integer - yPos : integer
Methods: + Constructor (inXPos : integer, inYPos : integer) + getXPos : integer + getYPos : integer + distanceFromPos (pos2: Position) : double + toString : string

NB: The distanceFromPos method will be implemented in a later question.

- 2.1 Write code to create a new class called **Position**. (1)
- 2.2 Write code to add the two fields **xPos** and **yPos** to the class. Ensure that these fields cannot be accessed outside of this class. (2)
- 2.3 Write code to create a **constructor** method that accepts the values for the two fields **xPos** and **yPos** as parameters. These parameters should be used to set the values of the two fields of the **Position** class. (2)
- 2.4 Write code to create an accessor method of the **xPos** field of the Position class. This method should be named appropriately and return the correct type and value for the **xPos** field. (1)
- 2.5 Write code to create an accessor method of the **yPos** field of the Position class. This method should be named appropriately and return the correct type and value for the **yPos** field. (1)
- 2.6 Write code to create a **toString** method that will return a string representing the x and y values. The format is as follows:

"Co-ordinates:<space>(" xPos ";" <space> yPos ")"

Example:

Co-ordinates: (33; 77) (2)

QUESTION 3

14 MARKS

Use the class diagram below to create a new class called **TrainingDummy**. This class will be used to create objects that will store the details of each training dummy. The diagram below indicates the fields and methods that are required.

TrainingDummy	
Properties:	
-	name : string
-	posi : Position
-	health: integer
Methods:	
+	Constructor (inName : string, inPosi : Position, inHealth : integer)
+	getPosi : Position
+	getName : string
+	takeDamage (integer inDamage)
+	distanceFromPosi (posi2: Position) : double
+	toString : string

NB: The distanceFromPosi method will be implemented in a later question.

- 3.1 Write code to create a new class called **TrainingDummy**. (1)
- 3.2 Write code to add the three fields to the class. Ensure that these fields cannot be accessed outside of this class. (3)
- 3.3 Write code to create a **constructor** method that accepts the details of the **TrainingDummy** class as parameters. These parameters should be used to set the values of the three fields of the **TrainingDummy** class. The **posi** is stored as a **Position** object (3)
- 3.4 Write code to create an accessor method of the **name** field of the **TrainingDummy** class. This method should be named appropriately and return the correct type and value for the **name** field. (1)
- 3.5 Write code to create an accessor method of the **posi** field of the **TrainingDummy** class. This method should be named appropriately and return a **Position** object. (1)
- 3.6 Write code to create an accessor method of the **health** field of the **TrainingDummy** class. This method should be named appropriately and return the correct type and value for the **health** field.
- 3.7 Write code to create a **takeDamage** method. This method should take in an integer value as a parameter. The **health** field should be reduced by the value of the parameter. If the **health** field is reduced to a value below zero, the health should be set to zero. (2)
- 3.8 Write code that will create a **toString** method that will return a string. The string should be returned in the following format:

```
name <space> "located at" <space> posi <space> "with health"
<space> health
```

Example:

Training Dummy 1 located at Co-ordinates: (11; 94) with health 93 (2)

QUESTION 4

13 MARKS

You are required to write code to create a new class called **Superhero**. This class will store the details of each superhero.

- 4.1 Write code to create a new class called **Superhero**. (1)
- 4.2 The **Superhero** class must store details of each superhero's **name** (string), **posi** (Position), **health** (integer), **damage** (integer) and **range** (integer). You may reuse existing classes instead of redeclaring fields. All fields should be private. (2)
- 4.3 Write code to create a constructor method for this class that will initialise all fields of the class. (3)
- 4.4 Write code to create an accessor method of the **damage** field of the **Superhero** class. This method should be named appropriately and return the correct type and value for the **damage** field. (1)
- 4.5 Write code to create an accessor method of the **range** field of the **Superhero** class. This method should be named appropriately and return the correct type and value for the **range** field. (1)
- 4.6 Write an **isAvenger** method. This method should return true if the word "Avenger" occurs anywhere in the **name** field of the object, otherwise it should return false. (3)
- 4.7 Write code that will create a **toString** method that will return a string. The string should be returned in the following format:

```
name <space> "located at" <space> posi <space> "with health"
<space> health ", damage" <space> damage <space> "and range"
<space> range
```

Example:

Okoye located at Co-ordinates (33, 77) with health 100, damage 32
and range 14 (2)

QUESTION 5**22 MARKS**

- 5.1 Write code to create a new class called **FighterManager**. (1)
- 5.2 Write code to declare the following TWO arrays and TWO counter variables. All fields should be private.
- An array to store up to 100 **TrainingDummy** objects and a counter to keep track of the number of **TrainingDummy** objects.
 - An array to store up to 100 **Superhero** objects and a counter to keep track of the number of **Superhero** objects. (4)
- 5.3 Write code to create a public constructor method that will read in the contents of the **fighters.txt** file and perform the following:
- Each line of the text file contains one training dummy or one superhero.
 - Check if the file exists and display an appropriate message if it does not exist.
 - Read each line of the file.
 - Each line will either be a training dummy or a superhero.
 - Create the appropriate **Training Dummy** or **Superhero** object depending on the whether information for **damage** and **range** exists or not.
 - Add the **TrainingDummy** or **Superhero** object to the appropriate array. (12)
- 5.4 Write code that will create a public **toString** method. This method should return a list of all the **TrainingDummy** objects followed by a list of all **Superhero** objects, each on a separate line as a String. Use the object's **toString** methods created in question 3 and question 4. (5)

Example:

TrainingDummies:

Training Dummy 11 located at Co-ordinates: (94; 57) with health 93

Training Dummy 1 located at Co-ordinates: (80; 29) with health 30

Training Dummy 3 located at Co-ordinates: (51; 21) with health 166

Superheroes:

Okoye located at Co-ordinates: (33; 77) with health 100, damage 32 and range 14

Avenger Captain America located at Co-ordinates: (69; 17) with health 59, damage 17 and range 12

Ant-Man located at Co-ordinates: (3; 84) with health 156, damage 10 and range 17

QUESTION 6**3 MARKS**

- 6.1 Create a user interface called **TutorialApp** that will allow simple output. (1)
- 6.2 Declare and instantiate a **FighterManager** object at the appropriate place in the code. (1)
- 6.3 Write code that will display all the **TrainingDummy** and **Superhero** objects by calling the appropriate method in the **TutorialApp** code. (1)

QUESTION 7

19 MARKS

You are required to add code to the existing classes in order to calculate the distance between a **Superhero** and a **TrainingDummy**.

7.1 Add code to the **Position** class:

Create a public **distanceFromPosi** method in the **Position** class. This method must take in another **Position** as a parameter. The method must calculate the distance between the current object and the **Position** sent as a parameter. Use the following formula:

$$distance = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Where the current object has co-ordinates $(x_1; y_1)$ and the **Position** sent as a parameter has co-ordinates $(x_2; y_2)$. Return the distance as a double. (5)

7.2 Add code to the **TrainingDummy** class to determine the distance of a training dummy to a position.

Create a public **distanceFromPosi** method in the **TrainingDummy** class. This method must take a **Position** as a parameter. The method must return the distance between the **TrainingDummy** and the **Position** sent as a parameter by calling the appropriate method using the appropriate field of the **TrainingDummy** class. Return the distance as a double. (3)

7.3 Add code to the **FighterManager** class

Create a public **attack** method in the **FighterManager** class. This method must take in a **Superhero name** parameter as a String and return a String.

7.3.1 Add code to search through the **Superhero** array until the **Superhero** with the correct **name** is found. (2)

7.3.2 Once the correct name is found then:

For every **TrainingDummy** in the **TrainingDummy** array the code should:

- Determine the distance from the **TrainingDummy** to the **Superhero**.
- If the distance is LESS THAN OR EQUAL TO the **Superhero** range then
 - Call the **takeDamage** method on the **TrainingDummy**. Send the **Superhero**'s damage as a parameter.
 - Build up a String with the name of the **TrainingDummy** that took damage, how much damage it took and the new health of the **TrainingDummy**. (7)

You will receive marks for the efficiency of your code. You may add any additional methods you need to the **Superhero** class.

7.4 Add code to the **TutorialApp** class

Print out the damage done by **Avenger Captain America** to any training dummies within range. In this it will only be one dummy. The output should look like this:

Training Dummy 6 took 17 damage. New health is 30. (2)