



eduresourceza.com

[Insert School Badge Here]

SUBJECT	Information Technology	DATE	November 2021
GRADE	11 Practical	MARKS	150
EXAMINER		DURATION	3 Hours
		MODERATOR	Ms Kench

NAME	
CLASS	

1. This question paper consists of **10** 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. Ensure 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.
5. 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 for data validation.
6. 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.
7. 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.
8. 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.
9. 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.
10. You must save all your work regularly on the disk you have been given, or the disk space allocated to you for this examination. You should also create a backup of the original files before you start in case the original version is accidentally modified by your solution.
11. Make sure that your examination number appears as a comment in every program that you code as well as on every page of hard copy that you hand in.
12. Print a code listing of all the programs/classes/output files 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.
13. You should be provided with the following two folders (in bold) and files. These files are to be used as data for this examination. Note that the database files are provided in MS Access, JavaDB and MySQL format. Ensure that you can open the files with the packages that you will use to code your solutions to this examination.

Section A:

- TreasureHunt.accdb
- TreasureHunt.sql
- SQL Answer Sheet

Section B:

- Challenge.txt

TREASURE HUNT

SECTION A

Suzi Q has decided to have a treasure hunt for her birthday during COVID. Party guests, divided into teams, will need to ride around her neighbourhood and participate in certain challenges. After each completed challenge the guest sends her a message with video proof that the challenge has been completed by the team. Each challenge has been allocated points and a difficulty rating (either safe or dangerous). Since she emailed the challenges to her guests, the guests were able to start their first challenge at exactly 15:00 (3pm).

Using this data, she created a database called **TreasureHunt** with a table **tblChallenges**. In the table, the challenge name, team name, time, points and rating are stored.

Her challenges are described in the table below, with the corresponding code she has used in the table **tblChallenges**:

Challenge	Code
Take a group selfie	Selfie
Get a paw print of a dog	PawPrint
Sit in a pothole	Pothole
Swop an item of clothing with a stranger	Swop
Tell a joke in a lift full of people	Joke
Pose on the bonnet of a luxury car	Pose

tblChallenges in Design View

Field Name	Data Type
Name	Short Text
TeamName	Short Text
Time	Date/Time
Points	Number
Safety	Short Text

Sample of data in tblChallenges

Name	TeamName	Time	Points	Safety
PawPrint	Finding Nemo	15:00:00	15	Dangerous
Pothole	In Pursuit	15:00:00	10	Dangerous
PawPrint	Recovery Rebels	15:00:00	15	Dangerous
PawPrint	No Clue	15:00:00	15	Dangerous
PawPrint	Lost n Found	15:00:00	15	Dangerous
Selfie	Lost n Found	15:00:45	5	Safe
Selfie	Recovery Rebels	15:01:16	5	Safe
Pothole	Finding Nemo	15:12:10	10	Dangerous
Pothole	Lost n Found	15:12:23	10	Dangerous
Swop	In Pursuit	15:14:34	20	Safe
Selfie	Finding Nemo	15:15:45	5	Safe
Pothole	No Clue	15:19:27	10	Dangerous
Jokes	In Pursuit	15:21:41	15	Safe
Selfie	In Pursuit	15:21:57	5	Safe
Jokes	Lost n Found	15:22:16	15	Safe
Swop	No Clue	15:23:32	20	Safe
Pothole	Lost n Found	15:23:52	10	Dangerous
Selfie	No Clue	15:25:35	5	Safe
Jokes	Recovery Rebels	15:27:52	15	Safe
Pose	Lost n Found	15:30:29	10	Safe

QUESTION ONE

SQL

50 MARKS

- 1.1 Show a list of the teams participating, in alphabetical order without any duplicates. Your output should appear as follows:

TeamName
Finding Nemo
In Pursuit
Lost n Found
No Clue
Recovery Rebels

(5)

- 1.2 Display the name of the challenges that start with th-completed by the team called "In Pursuit". List the **TeamName** and **Name** fields. Your output should appear as follows:

TeamName	Name
In Pursuit	Pothole
In Pursuit	Swop
In Pursuit	Jokes
In Pursuit	Selfie
In Pursuit	Pose

(3)

- 1.3 Create a code for each entry in the table using the first 2 letters of the name of the challenge, the last 32 letters of the team name and the minutes of the time field. Display the **Name**, **TeamName**, **Time** and newly created **Code** field. The resulting first 10 records are shown below:

Name	TeamName	Time	Code
PawPrint	Finding Nemo	15:00:00	Paemo0
Pothole	In Pursuit	15:00:00	Pouit0
PawPrint	Recovery Rebels	15:00:00	Paels0
PawPrint	No Clue	15:00:00	Palue0
PawPrint	Lost n Found	15:00:00	Paund0
Selfie	Lost n Found	15:00:45	Seund0
Selfie	Recovery Rebels	15:01:16	Seels1
Pothole	Finding Nemo	15:12:10	Poemo12
Pothole	Lost n Found	15:12:23	Pound12
Swop	In Pursuit	15:14:34	Swuit14
Selfie	Finding Nemo	15:15:45	Seemo15

(6)

- 1.4 Find the name of the team that was out the longest. Display the **TeamName** and the **Time**. Your output should appear as follows:

TeamName	Time
Finding Nemo	16:23:42

(4)

- 1.5 Determine the number of challenges completed by each team. Name this generated field **NumberOfChallenges**. Your output should be displayed as follows:

TeamName	NumberOfChallenges
Finding Nemo	9
In Pursuit	5
Lost n Found	6
No Clue	6
Recovery Rebels	6

(6)

- 1.6 Determine the total points **that** were awarded for each challenge. Display the name of the challenge combined with the difficulty rating in brackets. Name the derived fields **Challenge** and **PointsAllocated**. Display the result in descending order of **PointsAllocated**.

Challenge	PointsAllocated
Jokes (Safe)	105
PawPrint (Dangerous)	75
Swop (Safe)	60
Pothole (Dangerous)	60
Pose (Safe)	50
Selfie (Safe)	30

(10)

- 1.7 Determine the number of challenges completed by each team. Only list the teams that have completed more than 5 challenges. Your output should appear as follows:

TeamName	NumChallenges
Finding Nemo	9
Lost n Found	6
No Clue	6
Recovery Rebels	6

(7)

- 1.8 Looking back at the photos, Suzi realises that the two selfies sent in by Finding Nemo are in fact the same photo. Delete the selfie entry from the team Finding Nemo taken at 15:36:09.

(6)

- 1.9 Suzi decides to change the point allocation for the Pothole challenge to 25, as it is dangerous. Update all the points fields for the Pothole records to be 25.

(3)

SECTION B

A text file Challenge.txt has been provided. A sample of the data is shown below. The format of the file is:

Challenge, Team name, Time, Points, Safety

```
PawPrint,Finding Nemo,15:00:00,15,2
Pothole,In Pursuit,15:00:00,10,2
PawPrint,Recovery Rebels,15:00:00,15,2
PawPrint,No Clue,15:00:00,15,2
PawPrint,Lost n Found,15:00:00,15,2
Selfie,Lost n Found,15:00:45,5,1
Selfie,Recovery Rebels,15:01:16,5,1
Pothole,Finding Nemo,15:12:10,10,2
Pothole,Lost n Found,15:12:23,10,2
Swop,In Pursuit,15:14:34,20,1
Selfie,Finding Nemo,15:15:45,5,1
Pothole,No Clue,15:19:27,10,2
Jokes,In Pursuit,15:21:41,15,1
Selfie,In Pursuit,15:21:57,5,1
Jokes,Lost n Found,15:22:16,15,1
Swop,No Clue,15:23:32,20,1
Pothole,Lost n Found,15:23:52,10,2
Selfie,No Clue,15:25:35,5,1
Jokes,Recovery Rebels,15:27:52,15,1
Pose,Lost n Found,15:30:29,10,1
Pothole,Recovery Rebels,15:31:11,10,2
```

Formatted: stbNormal

QUESTION TWO**21 MARKS**

Create a class diagram called **Challenge** to store the details of each challenge.

Challenge
- name : string - teamName : string - time : Time - points : integer - safety : integer + <u>DANGEROUSNUM</u> : integer = 2
+ Constructor(n:string, tn:string, t:Time, p:integer, s:integer) + getName() : string + getTeamName() : string + getTime() : Time + getPoints() : integer + getSafety() : integer + addBonus(p : integer) + toString() : string

- 2.1 Create a class called **Challenge**. (1)
- 2.2 Add fields to store the **name**, **teamName**, **time**, **points** and **safety** using the types indicated in the class diagram. (3)
- 2.3 Add a constant static field to set **DANGEROUS** to 2. (3)

- 2.4 Code a parametrised constructor to assign values to the fields **name**, **teamName**, **time**, **points** and **safety** fields. (4)
- 2.5 Add accessor methods for the **name**, **teamName**, **time**, **points** and **safety** fields. (2)
- 2.6 Add a method called **addBonus** to increase the points field by the value sent as the parameter **p**. (4)
- 2.7 Add a **toString** method to display the **name**, **teamName**, **time**, **points** and **safety** fields separated by tabs. (4)

Sample output:

PawPrint Finding Nemo 15:00 15 2 (4)

QUESTION THREE

36 MARKS

```

ChallengeArray
- Challenge [] cArr
- size : integer
+ Constructor()
+ sort()
+ changePoints()
- shiftLess(p : integer)
+ removeDups()
+ teamPoints() : string
+ toString() : string

```

Formatted: No underline

- 3.1 Create a class called **ChallengeArray**. (1)
- 3.2 Create two fields that will be accessible through the whole class:
- 3.2.1 An array of 50 **Challenge** objects called **cArr**. (3)
- 3.2.2 A field called **size** to record the number of elements in the array called **cArr**. (2)
- 3.3 Code a constructor method to read the data from the text file called **Challenge.txt** and instantiate the array of **Challenge** objects called **cArr**. The method must:
- Open the text file called **Challenge.txt** and display an error message if the file is not found.
 - Read in each line of the text file.
 - Separate each line into the relevant fields.
 - Convert the time field into a **Time** object.
 - Instantiate the next **Challenge** element in the array called **cArr**.
 - Increase the size field. (10)
- 3.4 Code a **toString** method to display the objects in the **ChallengeArray** called **cArr**, each on a separate line. (5)
- 3.5 Code a method called **sort** to arrange the **ChallengeArray** called **cArr** in alphabetical order of the team name. (8)
- 3.6 Code a method called **changePoints** to add 7 points to each of the challenges in the array called **cArr** if the challenge was considered dangerous. Use the static field called **DANGEROUS** you created in Question 2.3 and the **addBonus** method you coded in Question 2.6. (7)

QUESTION FOUR**8 MARKS**

4.1 Code a text-based user interface class called **ChallengeUI**. (1)

4.2 Add code to instantiate an object called **ca** of the **ChallengeArray** class. (1)

4.3 Add code to display the list of challenges. A sample of the output is shown below: (2)

```
PawPrint      Finding Nemo      15:00      15      2
Pothole In Pursuit      15:00      10      2
PawPrint      Recovery Rebels 15:00      15      2
PawPrint      No Clue 15:00      15      2
PawPrint      Lost n Found      15:00      15      2
Selfie Lost n Found      15:00:45      5      1
Selfie Recovery Rebels 15:01:16      5      1
Pothole Finding Nemo      15:12:10      10      2
Pothole Lost n Found      15:12:23      10      2
Swop In Pursuit      15:14:34      20      1
```

4.4 Add code to sort the array and display the sorted array. A sample of the output is shown below: (2)

```
PawPrint      Finding Nemo      15:00      15      2
Pothole Finding Nemo      15:12:10      10      2
Selfie Finding Nemo      15:15:45      5      1
Jokes Finding Nemo      15:35:53      15      1
Selfie Finding Nemo      15:36:09      5      1
Swop Finding Nemo      15:45:36      20      1
Pose Finding Nemo      15:51:08      10      1
Jokes Finding Nemo      16:10:53      15      1
PawPrint      Finding Nemo      16:23:42      15      2
Pothole In Pursuit      15:00      10      2
Jokes In Pursuit      15:21:41      15      1
Selfie In Pursuit      15:21:57      5      1
Pose In Pursuit      15:34:10      10      1
Swop In Pursuit      15:14:34      20      1
Pothole Lost n Found      15:12:23      10      2
Selfie Lost n Found      15:00:45      5      1
Jokes Lost n Found      15:22:16      15      1
Pothole Lost n Found      15:23:52      10      2
Pose Lost n Found      15:30:29      10      1
```

4.5 Add code to ~~change~~ add bonus points to the dangerous challenges and display the array after the adjustment. A sample of the output is shown below: (2)

```
PawPrint      Finding Nemo      15:00      22      2
Pothole Finding Nemo      15:12:10      17      2
Selfie Finding Nemo      15:15:45      5      1
Jokes Finding Nemo      15:35:53      15      1
Selfie Finding Nemo      15:36:09      5      1
Swop Finding Nemo      15:45:36      20      1
Pose Finding Nemo      15:51:08      10      1
Jokes Finding Nemo      16:10:53      15      1
PawPrint      Finding Nemo      16:23:42      22      2
Pothole In Pursuit      15:00      17      2
Jokes In Pursuit      15:21:41      15      1
Selfie In Pursuit      15:21:57      5      1
Pose In Pursuit      15:34:10      10      1
Swop In Pursuit      15:14:34      20      1
```

Pothole	Lost n Found	15:12:23	17	2
Selfie	Lost n Found	15:00:45	5	1
Jokes	Lost n Found	15:22:16	15	1
Pothole	Lost n Found	15:23:52	17	2
Pose	Lost n Found	15:30:29	10	1

QUESTION FIVE**18 MARKS**

- 5.1 In the **ChallengeArray** class, code the **shiftLess** method to remove an object in the **ChallengeArray** called **cArr**, given the position of the element sent as the parameter called **p**. (6)
- 5.2 Code a method called **removeDups** to delete any objects in the **ChallengeArray** called **cArr** where a team has done the same challenge twice. For example, the team called "Finding Nemo" took two selfies. The first occurrence must be kept, and the second must be deleted. Use the method called **shiftLess** you coded in Question 5.1. (10)
- 5.3 Add code in the **ChallengeUI** class to call the **removeDups** method and display the array after the duplicates have been removed. A sample of the output is shown below: (2)

Formatted: Font: Not Bold

PawPrint	Finding Nemo	15:00	22	2
Pothole	Finding Nemo	15:12:10	17	2
Selfie	Finding Nemo	15:15:45	5	1
Jokes	Finding Nemo	15:35:53	15	1
Swop	Finding Nemo	15:45:36	20	1
Pose	Finding Nemo	15:51:08	10	1
Pothole	In Pursuit	15:00	17	2
Jokes	In Pursuit	15:21:41	15	1
Selfie	In Pursuit	15:21:57	5	1
Pose	In Pursuit	15:34:10	10	1
Swop	In Pursuit	15:14:34	20	1
Pothole	Lost n Found	15:12:23	17	2
Selfie	Lost n Found	15:00:45	5	1
Jokes	Lost n Found	15:22:16	15	1
Pose	Lost n Found	15:30:29	10	1

QUESTION SIX**17 MARKS**

- 6.1 In the **ChallengeArray** class, code a typed method called **teamPoints** to determine the sum of the points for each team. (14)
- 6.2 In the **ChallengeUI** class, add code to call the **teamPoints** method and display the result under the heading "Team Points". Your output should appear as follows, with teams and their points in any order: (3)

```
Team Points
Finding Nemo      89
In Pursuit        67
Lost n Found      69
No Clue           89
Recovery Rebels   69
```