



eduresourceza.com

[Insert School Badge Here]

SUBJECT
GRADE
EXAMINER

Information Technology
11 Practical MEMO

DATE
MARKS
DURATION
MODERATOR

November 2021
150
3 Hours
Ms Kench

NAME	
CLASS	

TREASURE HUNT

SECTION A

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)

```
SELECT DISTINCT TeamName
FROM tblChallenges
ORDER BY TeamName;
```

- 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)

```
SELECT TeamName, Name
FROM tblChallenges
WHERE TeamName="In Pursuit";
Accept WHERE TeamName LIKE "Th*"
Accept WHERE LEFT(TeamName,2) = "Th"
```

- 1.3 Create a code for each entry in the table using the first 2 letters of the name of the challenge, the last 2 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:

Formatted: Font: (Default) Courier New, Bold

Formatted: Font: (Default) Courier New, Bold

Formatted: Font: (Default) Courier New, Bold

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

SELECT [Name], TeamName, Time, ✓ LEFT([name],2) ✓ & ✓
 RIGHT(TeamName,3 or 2) ✓ & MINUTE(Time) ✓ As Code ✓
 FROM tblChallenges

Accept MID(time,4,2) to extract minutes.

MySQL:

SELECT Name, TeamName, Time, ✓ CONCAT(✓ (LEFT([name],2) ✓ ,
 RIGHT(TeamName,3 or 2) ✓ , MINUTE(Time) ✓) As Code ✓
 FROM tblChallenges

Formatted: Font: Bold

(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)

SELECT TOP 1 ✓ TeamName, Time ✓
 FROM tblChallenges

ORDER BY Time ✓ DESC ✓;

Alternate:

SELECT TeamName, Time ✓
 FROM tblChallenges

WHERE Time ✓ = (SELECT max(time) ✓ FROM tblChallenges) ✓

Formatted: Font: Bold

- 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)

```
SELECT TeamName, Count(*) AS NumberOfChallenges
FROM tblChallenges
GROUP BY TeamName;
```

- 1.6 Determine the total points 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)

```
SELECT [Name] & " (" & [Safety] & ")" AS Challenge, Sum(Points)
AS PointsAllocated
FROM tblChallenges
GROUP BY [Name] & " (" & [Safety] & ")"
ORDER BY Sum(Points) DESC
```

MySQL:

```
SELECT CONCAT (Name, ' (' & Safety & ')') AS Challenge,
Sum(Points) AS PointsAllocated
FROM tblChallenges
GROUP BY Challenge
ORDER BY PointsAllocated DESC
```

Formatted: Font: Bold

Formatted: Font: Bold

- 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)

```
SELECT [tblChallenges].[TeamName], count(*) AS NumChallenges
FROM tblChallenges
GROUP BY tblChallenges.TeamName
HAVING Count(tblChallenges.Name) > 5
Accept >= 4
```

MySQL:

```
SELECT TeamName, count(*) AS NumChallenges
FROM tblChallenges
```

Formatted: Font: (Default) Courier New, Bold

Formatted: Font: Bold

GROUP BY TeamName
HAVING NumChallenges >5

Formatted: Font: (Default) Courier
New

- 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)

DELETE *
 FROM tblChallenges
 WHERE TeamName="Finding Nemo"
 AND Name="Selfie"
 AND Time>#15:36:00#;
They MUST look for all the conditions mentioned in the question.
Although the time is unique they need to look at the mark allocation and read the question.
In Access Time=#15:36:09# does not work?
1 mark for either AND

Formatted: Font: Bold

- 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.

UPDATE tblChallenges
 SET tblChallenges.Points = 25
 WHERE Name="Pothole";

(3)

SECTION B

OOP

100 MARKS

//2.1 [1]

public class Challenge✓ Do not penalise for spelling errors. As long as they consistently use the same identifier name and the identifier has some resemblance to the required word.

Formatted: Font: Bold

{

//2.2 [3]

private✓ String ✓name; ✓
 private String teamName;
 private LocalTime time;
 private int points;
 private int safety; Penalise -1 for any extra fields.

Formatted: Font: Bold

//2.3 [3]

public static✓ final int DANGEROUS✓ = 2; ✓

Formatted: Font: Bold

//2.4 [4]

public Challenge✓(String n, String tn, LocalTime t, int p, int s)✓ MUST use these parameters in the order of the class diagram

{
 name = n; ✓✓
 teamName = tn;
 time = t;
 points = p;
 safety = s;
 }

//2.5 [2] ✓ One mark for correct names, ✓ One mark for correct return types,

public String getName() The order of the methods should be in the same as the class diagram but do not penalise unless you have emphasized it.

Formatted: Font: Bold

{
 return name;
 }

public String getTeamName()
 {
 return teamName;
 }

public LocalTime getTime()
 {
 return time;
 }

public int getPoints()
 {
 return points;
 }

public int getSafety()
 {
 return safety;
 }

//2.6 [4]

public void✓ addBonus(int p) ✓ Must be p

```

    {
        points ✓ = points + p; ✓ accept points += p;
    }
//2.7 [4]
@Override
public String toString() ✓ The format of the output must be exact.
{
    return ✓ name + "\t" ✓ + teamName + "\t" + time + "\t" + points
+ "\t" + safety; ✓ Cannot accept multiple spaces in place of tab
}
}

//3.1 [1]
public class ChallengeArray ✓
{
//3.2.1 [3]
    private ✓ Challenge[] cArr = new Challenge ✓ [50]; ✓
//3.2.2 [2]
    private int ✓ size ✓ = 0; Penalise -1 for any extra fields if not
penalised before. The field size does not need to be initialised to 0.
//3.3 [10]
    public ChallengeArray() Accept if they sent the filename as a
parameter but check they sent the correct file name in the method
call.
Penalise -1 if they throw the exception even if they catch it later.
    {
        try
        {
            Scanner inFile = new Scanner(new File("Challenge.txt")); ✓
            String line, name, team, time;
            int points, safe;
            LocalDateTime t;

            while (inFile.hasNext()) ✓
            {
                line = inFile.nextLine(); ✓
                Scanner sc = new Scanner(line).useDelimiter(","); ✓

                name = sc.next(); ✓
                team = sc.next();
                time = sc.next();

Accept the use of the split method to separate fields. 2 marks are
allocated to splitting fields.
                t = LocalDateTime.parse(time,
DateTimeFormatter.ofPattern("HH:mm:ss")); ✓✓
                points = sc.nextInt();
                safe = sc.nextInt(); If they used the text file with
the words "Safe" and "Dangerous" they would need an if statement to
convert the words to a 1 or 2. There are no marks allocated to this.
Penalise -1 if they make no attempt to convert, otherwise leave marks
as is even if the attempt has errors. As long as you mark
consistently for your school.
            }
        }
    }
}

```

Formatted: Font: Bold

Formatted: Font: Bold

Formatted: Font: Bold

Formatted: Font: Bold

Formatted: Font: Bold

Formatted: Font: Bold

```

        cArr[size] = new Challenge(name, team, t, points,
safe); ✓

        System.out.println(cArr[size]);
        size++; ✓
        sc.close();
    }
    inFile.close();
} catch (FileNotFoundException ex)
{
    System.out.println("File not found"); ✓ Only penalise -1 if
no message
}
}
//3.4 [5]
public String toString() ✓
{
    String temp = "";
    for (int i = 0; i < size; i++) ✓✓
    {
        temp = temp + cArr[i] + "\n"; ✓ The "\n" could be in the
toString of the Challenge class.
    }
    return temp; ✓
}
}
//3.5 [8]
public void sort() Accept any correct sort method that orders
according to TeamName.
{
    for (int i = 0; i < size - 1; i++) ✓✓ Make sure the start and
end values for both loops are exactly as shown. -1 for each error even
if it works. The sorts are algorithms that should be learnt.
    {
        for (int j = i + 1; j < size; j++) ✓✓
        {
            if
(cArr[i].getTeamName().compareTo(cArr[j].getTeamName()) > 0) ✓✓
            {
                Challenge temp = cArr[j]; ✓✓
                cArr[j] = cArr[i];
                cArr[i] = temp;
            }
        }
    }
}
//3.6 [7]
public void changePoints()
{
    for (int i = 0; i < size; i++) ✓
    {
        if (cArr[i].getSafety() ✓ == ✓ Challenge.DANGEROUS ✓)
        {
            cArr[i].addBonus ✓ (7 ✓);
        }
    }
}

```

Formatted: Font: Bold

Formatted: Font: Bold

```

    }
}
//5.1 [6]
private void shiftLess(int p) ✓
{
    for (int i = p; i < size - 1; i++) ✓✓
    {
        cArr[i] = cArr[i + 1]; ✓✓
    }
    size--; ✓
}

//5.2 [10] Accept any correct method that removes the duplicates
1 mark - method header
2 marks - processing the array with any valid loop
3 marks - comparing next element taking advantage of the sorted array
3 marks - comparing elements
1 mark - calling shiftLess with correct parameter
public void removeDups() ✓
{
    int j = 0;
    for (int i = 0; i < size - 1; i++) ✓✓
    {
        j = i + 1; ✓
        while (j < size) ✓
        {
            if (cArr[i].getName().equals(cArr[j].getName()) ✓ &&
✓ cArr[i].getTeamName().equals(cArr[j].getTeamName())) ✓
            {
                shiftLess(j); ✓
            } else
            {
                j++; ✓
            }
        }
    }
}

//6.1 [14]
public String teamPoints() ✓
{
    1 mark - header
    1 mark - return String
    6 marks - total up the points
    • 2 marks - initialise the sum and teamName
    • 2 marks - add team points to sum
    • 2 marks - move to next team
    • -3 marks if they hard coded each team
    2 marks - any loop
    2 marks - check if array element is the same team
    2 marks - build the string
    String temp = "";
    int i = 1;
    int sum = cArr[0].getPoints(); ✓
    String team = cArr[0].getTeamName(); ✓
}

```

Formatted: Font: Bold

Formatted: Font: Bold

Formatted: Bulleted + Level: 1 +
Aligned at: 0,25" + Indent at: 0,5"

Formatted: Font: Bold

```

while (i < size) ✓
{
    if (cArr[i].getTeamName().equals(team)) ✓✓
    {
        sum = sum + cArr[i].getPoints(); ✓✓
    } else
    {
        temp = temp + team + "\t" + sum + "\n"; ✓
        sum = cArr[i].getPoints(); ✓
        team = cArr[i].getTeamName(); ✓
    }
    i++; ✓
}
temp = temp + team + "\t" + sum + "\n"; ✓
return temp; ✓
}

//4.1 [1]
public class ChallengeUI ✓
{
    public static void main(String[] args)
    {
//4.2 [1]
        ChallengeArray cpa = new ChallengeArray(); ✓
//4.3 [2]
        System.out.println ✓ (cpa); ✓
accept System.out.println (cpa.toString());
//4.4 [2]
        cpa.sort(); ✓
        System.out.println("\n\nAfter sort\n" + cpa); ✓ Do not penalise
if they do not add a heading
//4.5 [2]
        cpa.changePoints(); ✓
        System.out.println("\n\nAfter Points Added\n" + cpa); ✓ Do not
penalise if they do not add a heading

//5.3 [2]
        cpa.removeDups(); ✓
        System.out.println("\n\nAfter Duplicates Removed\n" + cpa); ✓
Do not penalise if they do not add a heading

//6.2 [3]
        System.out.println ✓ ("Team Points\n" ✓ + cpa.teamPoints()); ✓
cpa.teamPoints() must be called as a typed method.
    }
}

```

Formatted: Font: Bold