Mock_Quiz - 2 Solution

Question 1 [2 Marks]

Statement

Let **L** be a non-empty list of integers, and **D** be a non-empty dictionary which are given below:
L = [8, -4, 10, 4, -6]
D = {1: {"Name": "John", "Age": 12, "Gender": 'M'}}
Match the following expressions on the left side with the appropriate values on the right side.

Left side	**Right side**
a. **member**(L, 10)	1. 12
b. **isKey**(**D**, "Name")	2. 10
c. **D**[1]["Age"]	3. [1]
d. **first**(L) + **last**(**init**(**L**))	4. False
e. **first**(**rest**(**rest**(**L**)))	5. True
f. **keys**(**D**)	6. Invalid expression

Options

(a)

a - (5), b - (5), c - (6), d - (3), e - (2), f - (6)

(b)

a - (5), b - (4), c - (1), d - (1), e - (2), f - (3)

(c)

a - (5), b - (4), c - (6), d - (1), e - (1), f - (6)

(d)

a - (5), b - (5), c - (1), d - (1), e - (2), f - (3)

(e)

a - (5), b - (6), c - (1), d - (1), e - (2), f - (3)

Answer

(b)

Solution

Solution:
a. List L contains 10 therefore, member function will return True.

b. Dictionary D is a nested dictionary, at first level it has only one key which is 1. And, then at second level it have “Name”, “Age” and “Gender”. Therefore, isKey function will return False because D does not contain “Name”, D[1] contains “Name”.

c. D[1][“Age”] will first look for key 1 in dictionary D then inside that it will search for key “Age” and find its value which is 12.

d. first(L) is 8, init(L) is [8, -4, 10, 4] and last of this list will be 4. Therefore, the answer is 8 + 4 = 12.

e. rest(L) will return [-4, 10, 4, -6], once again rest of this list will be [10, 4, -6] and then first of this list will be 10.

f. keys(D) returns the list of all keys in a dictionary D. As we saw earlier, dictionary D has only one key at first level therefore, the answer will be [1].

Click here to see the video solution for this question created by student

Click here to share your feedback regarding this solution

Question 2 [2 Marks]

Statement

What will the value of **outList** be at the end of the given pseudocode?

```
N = [1, 2, 3, 4, 5, 6, ....., 49, 50]
A = someList(N)
B = someList(rest(rest(N)))
outList = []
foreach Y in A {
    foreach Z in B {
        if(Z == Y){
            outList = outList ++ [Y]
        }
    }
}

Procedure someList(X)
    outlist = [ ], newList = X
    while(length(newList) > 0){
        outlist = outlist ++ [first(newList)]
        newList = rest(rest(newList))
    }
    return(outlist)
End someList
```

Options

(a)

[1, 2, 3, 4, 5, 6, ..., 49, 50]

(b)

[1, 3, 5, 7, ..., 49]

(c)

[2, 4, 6, 8, ...,50]

(d)

[3, 5, 7, ..., 49]

Answer

(d)

Solution

Procedure someList accepts a list as an input and returns another list as output containing all the alternate elements in the original list starting from from the first element. So, the value of A will [1, 3, 5, ..., 47, 49].

Now B = someList(rest(rest(N))), which means B = someList(rest(rest([3, 4, 5, 6, ..., 49, 50])))

Therefore, value of B will be [3, 5, ..., 47, 49].

In the nested foreach loop we are finding the common elements from lists A and B. As we know A holds all odd numbers between 1 and 50 whereas B holds all odd numbers from 3 to 50 so, the intersection will be a list all odd numbers from 3 to 50.

Click here to see the video solution for this question created by student

Click here to share your feedback regarding this solution

Question 3 [4 Marks]

Statement

Two trains are called “Opposite Trains” if they stop at the same set of stations but in the reverse order. **isOpposite**(**N1**, **N2**) returns True if trains with train numbers **N1** and **N2** are “Opposite Trains” and False otherwise. **trains** is a dictionary with train number as key mapped to a list of stations which that train runs through. For example, trains = { 12281: [“Bhubaneswar”, “Balasore”, “Adra”, “Varanasi”, “Kanpur”, “New Delhi”],.....}.

In this example, the train with train number 12281 starts from Bhubaneswar and reaches New Delhi via Balasore, Adra, Varanasi, and Kanpur in the order. Choose the correct code fragments to complete the procedure. It is a Multiple Select Question (MSQ).

```
Procedure isOpposite(N1, N2)
    L1 = trains[N1]
    L2 = trains[N2]
    if(length(L1) ≠ length(L2)){
        return(False)
    }
    while(L1 ≠ [ ] and first(L1) == last(L2)){
        L1 = rest(L1)
        L2 = init(L2)
    }
    **********************
    * Fill the code *
    **********************
End isOpposite
```

Options

(a)

```
if(L1 == [ ]){
    return(True)
}
else{
    return(False)
}
```

(b)

```
if(L1 == [ ]){
    return(False)
}
else{
    return(True)
}
```

(c)

```
if(L2 == [ ]){
    return(False)
}
else{
    return(True)
}
```

(d)

```
if(L2 == [ ]){
    return(True)
}
else{
    return(False)
}
```

Answer

(a), (d)

Solution

Lists L1 and L2 contains all the stations from N1 and N2 respectively. If the length of these lists is equal then only we proceed to while loop. In while loop we are comparing the first element of L1 and the last element of L2, if they are equal and L1 is not empty then we will continue the execution of while loop. Inside this loop we are shrinking the lists L1 and L2 from opposite ends by computing rest(L1) and init(L2) respectively. At the end of the execution of while loop, lists L1 and L2 should be empty and in such a case the procedure isOpposite will return True and False otherwise.

Click here to see the video solution for this question created by student

Click here to share your feedback regarding this solution

Question 4 [4 Marks]

Statement

Let **trains** be a dictionary with train number as key mapped to a list of stations which that train runs through. For example, **trains** = { 12281: [“Bhubaneswar”,“Balasore”, “Adra”, “Varanasi”, “Kanpur”, “New Delhi”],.....}. In this example, the train with train number 12281 starts from Bhubaneswar and reaches New Delhi via Balasore, Adra, Varanasi, and Kanpur.

At the end of execution of the code below, **L** stores the names of stations through which the maximum number of trains pass. Choose the correct fragment to complete the pseudocode.

```
stns = { },
N = 0, L = [ ]
foreach X in keys(trains){
    stns = updateDictionary(stns, X)
}
foreach Y in keys(stns){
    **********************
    * Fill the code      *
    **********************
}

Procedure updateDictionary(D, Z)
    foreach A in trains[Z]{
        if(not isKey(D, A)){
            D[A] = 1
        }
        else{
            D[A] = D[A] + 1
        }
    }
    return(D)
End updateDictionary
```

Options

(a)

```
if(stns[Y] == N){
    L = L ++ [Y]
}
if(stns[Y] > N){
    L = [Y]
    N = stns[Y]
}
```

(b)

```
if(stns[Y] > N){
    L = L ++ [Y]
    N = stns[Y]
}
if(stns[Y] == N){
    L = [Y]
}
```

(c)

```
if(stns[Y] == N){
    L = L ++ [Y]
    N = stns[Y]
}
else{
    L = [Y]
}
```

(d)

```
if(stns[Y] > N){
    L = L ++ [Y]
    N = stns[Y]
    exitloop
}
if(stns[Y] == N){
    L = [Y]
}
```

Answer

(a)

Solution

The procedure updateDictionary accepts two parameters first, a dictionary with station names as keys and the number of trains visiting that station as values and second is a train number. This procedure updates the input dictionary by iteration over all the stations listed on the train card whose train number it receives. And, the same updated dictionary is returned back as an output. Now, we are supposed to find the list of all the stations who have the highest value associated with it. Therefore, option a is correct where we are appending the station name to the list if we find another entry with same value. On the other hand if we find a new station with higher value then we are updating the variable N and resetting the list L.

Question 5 [5 Marks]

Statement

The following pseudocode is executed using the "Shopping Bills" dataset. At the end of the execution, **countPairs** should store the list of triples such that every triple has two items and the number of times both the items have been purchased together. For example if "Milk" and "Bread" have been purchased together three times, then the triple would be ["Milk", "Bread", 3].

The procedure **getPair**(**X**) returns the list of pairs of items purchased in card **X**. For example if "Milk", "Bread", and "Sugar" are present in the card **X**, **getPairs**(**X**) will return [["Milk", "Bread"], ["Milk", "Sugar"], ["Bread", "Sugar"]].

Choose the correct code fragments to complete the pseudocode.

```
countPairs = [], xpairs = [], pairList = []
while(Pile 1 has more cards){
    Read the top card X from Pile 1
    xpairs = getPairs(X)
    pairList = pairList ++ xpairs
    Move X from Pile 1 to Pile 2
}

while(length(pairList) > 1){
    restList = [], firstPair = [], tripple = []
    pairCount = 0, item1 = "None", item2 = "None"
    firstPair = first(pairList)
    item1 = first(firstPair)
    item2 = last(firstPair)
    foreach pair in rest(pairList){
        if(first(pair) == item1 or last(pair) == item1){
            ***************************
            **      FILL THE CODE      **
            ***************************
        }
        else{
            restList = restList ++ [pair]
        }

    }
    tripple = [item1, item2, pairCount]
    countPairs = countPairs ++ [tripple]
    pairList = restList
}
```

Options

(a)

```
if(first(pair) == item2 or last(pair) == item2){
    pairCount = pairCount + 1
}
else{
    restList = restList ++ [pair]
}
```

(b)

```
if(first(pair) == item1 or first(pair) == item2){
    pairCount = pairCount + 1
}
else{
    restList = restList ++ [pair]
}
```

(c)

```
if(last(pair) == item2 and first(pair) == item2){
    pairCount = pairCount + 1
}
else{
    restList = restList ++ [pair]
}
```

(d)

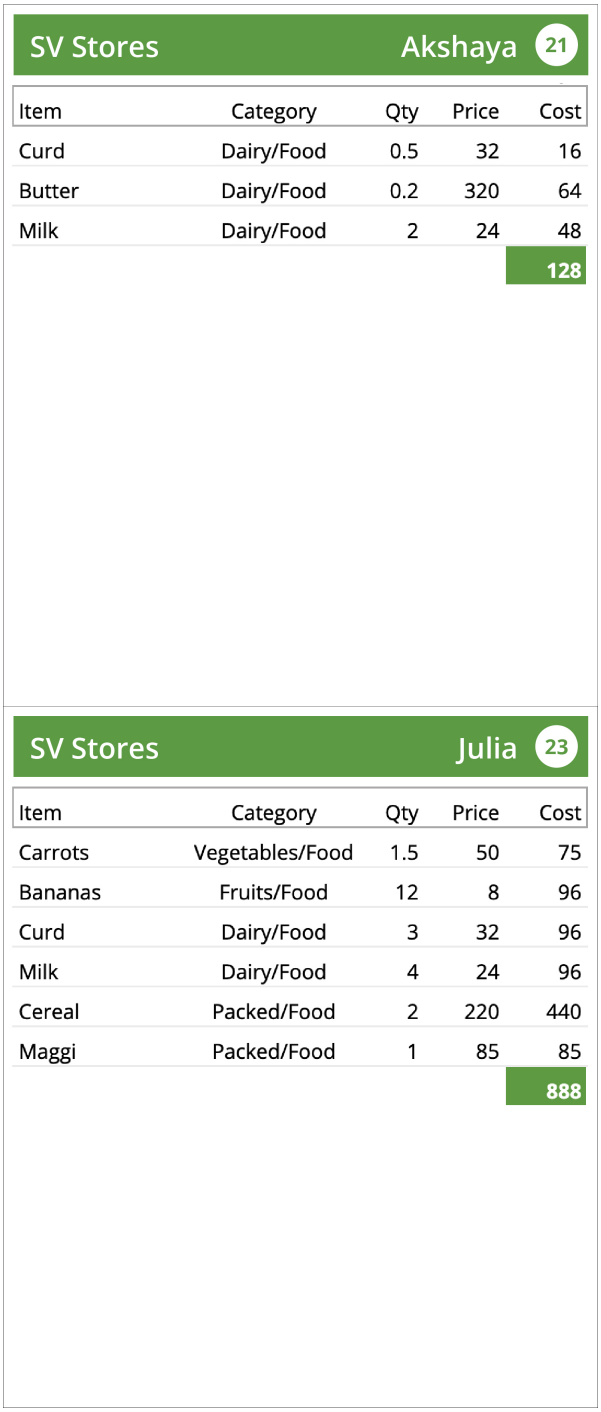
```
if(first(pair) == item1 and last(pair) == item1){
    pairCount = pairCount + 1
}
else{
    restList = restList ++ [pair]
}
```

Answer

(a)

Solution

To understand better the above question, let us take any two cards from the dataset.



SV Stores — Akshaya — 21

Item	Category	Qty	Price	Cost
Curd	Dairy/Food	0.5	32	16
Butter	Dairy/Food	0.2	320	64
Milk	Dairy/Food	2	24	48
				128

SV Stores — Julia — 23

Item	Category	Qty	Price	Cost
Carrots	Vegetables/Food	1.5	50	75
Bananas	Fruits/Food	12	8	96
Curd	Dairy/Food	3	32	96
Milk	Dairy/Food	4	24	96
Cereal	Packed/Food	2	220	440
Maggi	Packed/Food	1	85	85
				888

Line4: xpairs stores the list of pairs of item purchased in card X. This means xpairs stores the list of pairs of items purchased together in the Card X. If card X is the first card shown above then

xpairs = [["Curd", "Butter"], ["Curd", "Milk"], ["Butter", "Milk"]]

Line 5: This list of pairs gets appended in pairList. For the first time pairList will be the list of pairs of item seen in first card. In this example pairList = [["Curd", "Butter"], ["Curd", "Milk"], ["Butter", "Milk"]].

Again for the second card,

xpairs = [["Carrots", "Bananas"], ["Carrots", "Curd"], ["Carrots", "Milk"], ["Carrots", "Cereal"], ["Carrots", "Maggi"], ["Bananas", "Curd"], ["Bananas", "Milk"], ["Bananas", "Cereal"], ["Bananas", "Maggi"], ["Curd", "Milk"], ["Curd", "Cereal"], ["Curd", "Maggi"], ["Milk", "Cereal"], ["Milk", "Maggi"], ["Cereal", "Maggi"]]

Now again this list will get appended in pairList and pairList will become

pairList = [["Curd", "Butter"], ["Curd", "Milk"], ["Butter", "Milk"], ["Carrots", "Bananas"], ["Carrots", "Curd"], ["Carrots", "Milk"], ["Carrots", "Cereal"], ["Carrots", "Maggi"], ["Bananas", "Curd"], ["Bananas", "Milk"], ["Bananas", "Cereal"], ["Bananas", "Maggi"], ["Curd", "Milk"], ["Curd", "Cereal"], ["Curd", "Maggi"], ["Milk", "Cereal"], ["Milk", "Maggi"], ["Cereal", "Maggi"]]

This means pairList is the list of pairs of items purchased together from the whole dataset.

Now countPair is supposed to store the triples. To find the same pair the usual procedure is taken. Only one problem is here that the order of the items in pairs is not fixed. For example ["Bananas", "Curd"] and ["Curd", "Bananas"] are same pairs with respect to the pair of items but ["Bananas", "Curd"] $\neq$ ["Curd", "Bananas"] with respect to the list datatypes. Therefore we need to check either the first item of first pair is same as either the first or the second item of second pair. And same for the second item of the first pair should be checked. If matches for both then increment the pairCount.

In line 16, the first item of the first pair is being checked. If it is True then the second item should be checked. If the second items also matches then pairCount should be incremented. If does not matches then the pair should be appended to the restList.

After every run, restList stores the pairs of items which are found to be mismatched with the first pair of pairList.

Line 26: tripple stores the two items and the number of times these items have been purchased together.

Line 27: This triple is being appended to the final list (countPairs).

Click here to see the video solution for this question created by student

Click here to share your feedback regarding this solution

Questions - (6 - 8)

Question 6 [4 Marks]

Statement

The given pseudocode is executed using the "Scores" dataset. At the end of the execution of given pseudocode, choose the correct option(s)? It is a Multiple Select Question (MSQ).

```
topper = findTop(Table 1)
Procedure findTop(Table 1)
    max = 0, top = []
    while(Table 1 has more rows){
        Read the first row X from Table 1
        if(X.Total > max){
            max = X.Total
            top = [X.SeqNo, max]
        }
         Move X to Table 2
    }
    Move all rows from Table 2 to Table 1
    return(top)
End findTop
```

Options

(a)

topper is a list of two elements.

(b)

first(topper) represents the sequence number of one of the students who have scored the highest total marks

(c)

first(topper) is the list of the sequence numbers of the students who have scored the highest total marks

(d)

last(topper) is the highest total mark from the Table 1

Answer

(a), (b), (d)

Solution

The following pseudocode is executed on the "Scores" dataset. In the pseudocode a procedure called **findTop** is defined which takes a table as input. In the procedure while loop iterates through each rows of the Table 1. Inside while loop, total marks of each card **X** is compared with **max** in if-block. If the total marks of **X** is more than **max** then **X**.Total is stored in variable **max**. And also list of sequence number and **max** is stored in list **top**. Now let us investiagte each options:

Option (a): It is a true statement.

Option (b): The **first(topper)** will represents the sequence number of one of the student who got highest total marks. Therefore this is also a correct statement.

Option (c): This is incorrect statement because **first(topper)** will represent an integer (sequence no) not the list.

Option (d): This is correct statement since **last(topper)** will holds the highest total marks from the Table 1.

The correct options are (a), (b), and (d).

Click here to see the video solution for questions 6-8 created by student

Click here to share your feedback regarding this solution

Question 7 [3 Marks]

Statement

Consider the procedure **findTop** mentioned in question 6. The following pseudocode is executed using the "Scores" dataset. Let **removeRow**(**Table, z**) deletes the row of **Table** with sequence number **z**. What will **marksList** represent at the end of the execution of given pseudocode?

```
topper = [], marksList = []
while(Table 1 has more rows){
    topper = findTop(Table 1)
    removeRow(Table 1, first(topper))
    marksList = marksList ++ [topper]
}
```

Options

(a)

List of sequence numbers in ascending order based on the total marks

(b)

List of sequence numbers in descending order based on the total marks

(c)

List of pairs of sequence number and total marks in descending order based on the total marks

(d)

List of pairs of sequence number and total marks in ascending order based on the total marks

Answer

(c)

Solution

From the previous question we know that procedure **topper** will will store the sequence number and highest total marks of one of the student. In the given pseudocode, after finding the sequence number of one of the student who got highest total marks is removed from the Table 1 using procedure **removeRow**. Then **topper** as a list is appended to another list called **marksList**. And this process continues till Table 1 has no rows. So list **markList** will store the pairs of sequence number and total marks in descending order based on the total marks. There may be more then one student whose total marks are same. In this situation their positions are in cosecutive to each other.

Therefore the correct option is (c).

Question-8 [4 Marks]

Statement

Consider the **marksList** created in question 7. What will **topList** represent at the end of the execution of the given pseudocode?

```
topList = [], counter = 1
max = last(first(marksList))
foreach pair in marksList{
    if(last(pair) != max){
        max = last(pair)
        counter = counter + 1
    }
    if(counter < 4){
        topList = topList ++ [first(pair)]
    }
}
```

Options

(a)

List of sequence numbers of students whose total marks is in the top three total marks

(b)

List of sequence numbers of three students whose total marks is in the top three total marks

(c)

List of pairs of the sequence number and the total marks of students whose total marks is in the top four total marks

(d)

List of pairs of the sequence number and the total score of students whose total marks is in the top three total marks

Answer

(a)

Solution

This question uses the information of list **markList**, created in previous question. In the pseudocode, a variable called **max** stores the highest total marks, evaluated from **last(first(markList)). forach** loop iterates through each element (which is a list) of **markList**. Inside foreach loop, **last(pair)** is comapred with the **max**. If value of **last(pair)** is not equal to **max** then **last(pair)** is stored in **max** and **counter** is incremented. Since counter is less than 4, **first(pair)** (sequnce number) is appended to list **topList**.

So if more than one student got highest total marks then his/her sequence number will be appended list **topList** without incrementing **counter**. The value of **max** will be updated for the second highest total marks and then the sequence number of student who got second highest total marks will be appended to list **topList**. For the case of second highest total marks **counter** is 2. So if-block ("if(**counter** < 4)") will allow to append till third highest total marks.

Hence topList will store list of sequence numbers of students whose total marks is in the top three total marks. Hence the correct answer is (a).

Question (9 - 11)

Statement

The given pseudocode is executed using a dataset having the same fields as the “Words” dataset, and contains the following words -

“I ordered this product from Gitark. I am very happy to share my review regarding this awesome product. It is not only nice to use, but also has a very cool look. I think this is the best product which can be bought in this price range. The only bad thing was the broken and ugly packaging. I have never seen packaging worse than this before.”

Consider the following information:

- **unique**(**L**) returns a list of unique elements of list L.
- **comNo**(**L1**, **L2**) returns the number of common elements in lists L1 and L2.
- **max**(**A**, **B**, **C**) returns the maximum
- Upper case and lower case are ignored when checking whether a word belongs to positiveList and negativeList.

```
positiveList = [“happy”, “awesome”, “nice”, “fine”, “best”, “cool”]
negativeList = [“dull”, “worse”, “ugly”, “hopeless”, “bad”]
posSen = 0, negSen = 0, neutSen = 0, L = [ ]

while(Table 1 has more rows){
    Read the first row X in Table 1
    L = L ++ [X.Word]
    if(X.Word has full stop){
        L = unique(L)
        posCount = comNo(positiveList, L)
        negCount = comNo(negativeList, L)
        if(posCount ≥ 2 or negCount ≥ 2){
            if (posCount > negCount) {
                posSen = posSen + 1
            }
            else{
                negSen = negSen + 1
            }
        }
        else{
            neutSen = neutSen + 1
        }
        L = [ ]
    }
}

commentType = max(posSen, negSen, neutSen)
```

Based on this information, answer the following questions

Question 9 [4 Marks]

Statement

At the end of execution, the value of **posSen** will be

Options

(a)

2

(b)

3

(c)

4

(d)

5

Answer

(a)

Click here to see the video solution for questions 9 - 11 created by student

Click here to share your feedback regarding this solution

Question 10 [4 Marks]

Statement

At the end of execution, the value of **neutSen** will be

Options

(a)

2

(b)

3

(c)

4

(d)

5

Answer

(b)

Question 11 [4 Marks]

Statement

At the end of execution, the value of **commentType** will be

Options

(a)

neutSen

(b)

posSen

(c)

negSen

Answer

(a)

Solution:

Every sentence in the given paragraph will be termed as either a positive sentiment or a negative sentiment or a neutral sentiment. This categorization is based on the number of positive and/or negative words in the sentence.

1. I ordered this product from Gitark.
   posCount = 0 and negCount = 0 therefore, neutSen is incremented to 1.
2. I am very happy to share my review regarding this awesome product.
   posCount = 2 and negCount = 0 therefore, posSen is incremented to 1.
3. It is not only nice to use, but also has a very cool look.
   posCount = 2 and negCount = 0 therefore, posSen is incremented to 2.
4. I think this is the best product which can be bought in this price range.
   posCount = 1 and negCount = 0 therefore, neutSen is incremented to 2.
5. The only bad thing was the broken and ugly packaging.
   posCount = 0 and negCount = 2 therefore, negSen is incremented to 1.
6. I have never seen packaging worse than this before.
   posCount = 0 and negCount = 1 therefore, neutSen is incremented to 3.
   Therefore, the final value of posSen will 2, neutSen will 3 and commentType will “neutSen”

Question (12 - 14)

Statement

The following pseudocode is executed using the "Olympics" dataset. At the end of the execution, **medalLists** should store the of number of players who have won all the three types of medals and have won at least one type of medal more than one time. Assume that the Table 1 is always gets restored after every iteration.

```
1  medalLists = 0
2
3  while(Table 1 has more rows){
4      Read the first row X in Table 1
5      medaLists = medaLists + qualified(medalDict, repeatDict, X.SeqNo)
6      Move X to Table 2
7  }
8
9  Procedure qualified(medalDict, repeatDict, n)
10     Medals = 0, repeated = False
11     Medals = len(keys(medalDict[n])
12     repeated = repeatDict[n]
13     if(repeated and (Medals == 3)){
14         return(1)
15     }
16     return(0)
17 End qualified
```

Explanation

From line 11, it is clear that medalDict is a dictionary. It is also clear that n is the key of medalDict and the value of medalDict[n] is again a dictionary because the keys function is being applied. Which means medalDict is a nested dictionary. Now Medals is an integer and it is the number of keys stored in medalDict[n]. From line 13, as Medals == 3 is being checked and the question was about three distinct medals then it is evident that the keys of medalDict[n] are "Gold", "Silver", and "Bronze". If the Medals is tracking the first condition mentioned in the question, then repeated is tracking the second condition mentioned in the question. As repeated is a Boolean data type then, it should be either True or False. The value of repeat is decided by the repeatDict[n].

As repeat is the tracker of second condition, which is either the player has won at least one medal more than one time or not, then MedalDict is a dictionary with the keys as sequence numbers mapped to True if the player has won the medal.

Question 12 [3 Marks]

Statement

Which of the following statement(s) is(are) true about **medalDict** based on the above pseudocode? It is a Multiple Select Question (MSQ).

Options

a)

medalDict is a procedure which accepts the sequence number of a player and returns True if the player has won all the three types of medals otherwise returns False.

(b)

medalDict is a dictionary with sequence numbers of players mapped to True if the player has won all the three types of medals otherwise mapped to False.

(c)

medalDict is a nested dictionary with sequence numbers of players mapped to a dictionary with medals as keys mapped to True if the player has won that medal.

(d)

medalDict is a dictionary with sequence numbers of players mapped to the list of distinct medals won by the player.

Answer

(c)

Click here to see the video solution for questions 12 - 14 created by student

Click here to share your feedback regarding this solution

Question 13 [3 Marks]

Statement

Which of the following statement(s) is(are) true about **repeatDict** based on the above pseudocode? It is a Multiple Select Question (MSQ).

Options

a)

repeatDict is a procedure which accepts the sequence number of a player and returns True if the player has won at least one type of medal more than one time.

(b)

repeatDict is a dictionary with sequence numbers of players mapped to the number of medals which the player has won more than one time.

(c)

repeatDict is a dictionary with sequence numbers of players mapped to True if the player has won at least one type of medal more than one time.

(d)

repeatDict is a dictionary with sequence numbers of players mapped to True if the player has won at least one type of medal more than one time otherwise mapped to False.

Answer

(d)

Question 14 [4 Marks]

Statement

Let **G**, **S**, and **B** be the lists of the sequence numbers of the players who have won Gold medal, Silver medal, and Bronze medal respectively. If **n** is the sequence number of a player, then choose the correct implementation of **medalDict**?

Options

a)

```
Procedure medalDict(n)
    count = 0
    if(member(G, n)){
        count = count + 1
    }
    if(member(S, n)){
        count = count + 1
    }
    if(member(B, n)){
        count = count + 1
    }
    return(count)
End nSub
```

(b)

```
Procedure medalDict(n)
    count = 0
    if(member(G, n)){
        count = count + 1
    }
    if(member(S, n)){
        count = count + 1
    }
    if(member(B, n)){
        count = count + 1
    }
    if(count == 3){
        return(True)
    }
    return(False)
End nSub
```

(c)

```
medalDict = {}
while(Table 1 has more rows){
    Read the first row X from Table 1
    count = 0
    if(member(G, X.SeqNo)){
        count = count + 1
    }
    if(member(S, X.SeqNo)){
        count = count + 1
    }
    if(member(B, X.SeqNo)){
        count = count + 1
    }
    if(count == 3){
        medalDict[SeqNo] = True
    }
    Move X to Table 2
}
```

(d)

```
medalDict = {}
while(Table 1 has more rows){
    Read the first row X from Table 1
    medalDict[SeqNo] = {}
    if(member(G, X.SeqNo)){
        medalDict[SeqNo]["Gold"] = True
    }
    if(member(S, X.SeqNo)){
        medalDict[SeqNo]["Silver"] = True
    }
    if(member(Bronze, X.SeqNo)){
        medalDict[SeqNo]["Bronze"] = True
    }
    Move X to Table 2
}
```

Answer

(d)