Week 1: Practice Assignment

Note: Different font styles or punctuation symbols are used as follows

I. Italic + Underline: Steps of the procedure
II. Bold: Variables
III. Italic: A field from dataset
IV. " ": A value from dataset or dataset name

1. What will the values of **A**, **B** and **C** represent at the end of execution of the given procedure on the "Scores" dataset?

   Step 1. Arrange all cards in a single pile called Pile 1
   Step 2. Maintain three variables **A**, **B**, **C** and initialize them to 0
   Step 3. If Pile 1 is empty then stop the iteration
   Step 4. Read the top card in Pile 1
   Step 5. If *Date of Birth* < "1 May" then increment **A**
   Step 6. If *Date of Birth* > "30 April" and *Date of Birth* < "1 September" then increment **B**
   Step 7. If *Date of Birth* > "31 August" then increment **C**
   Step 8. Move the current card to another pile called Pile 2 and repeat from step 3

   Answer: A=10, B=9, C=11 (Numerical input)

2. The following procedure counts the number of words from the "Paragraph Words" dataset where *part of speech* is "Noun" and *Letter count* is greater than or equal to 4. But the programmer may have made mistakes in one or more steps. Identify all such steps (if any). It is a Multiple Select Question (MSQ).

   Step 1. Arrange all cards in a single pile called Pile 1
   Step 2. Maintain a variable **count** and initialize it to 1
   Step 3. If Pile 1 is empty then stop the iteration
   Step 4. Read the top card in Pile 1
   Step 5. If *part of speech* is "Verb" and *Letter count* ≥ 4 then increment **count**
   Step 6. Move the current card to another pile called Pile 2 and repeat from step 3

   a. Step 1
   b. Step 2
   c. Step 3
   d. Step 4
   e. Step 5
   f. Step 6
   g. No mistake

3. The given procedure finds the average *total bill amount* of shopping bills from the "Shopping Bills" dataset. Fill in the blanks from the given choices and complete the procedure.

   Step 1. Arrange all cards in a single pile called Pile 1
   Step 2. Maintain two variables **count**, **sum** and initialize them to 0
   Step 3. If Pile 1 is empty then stop the iteration and start from step 8
   Step 4. Read the top card in Pile 1
   Step 5. ____
   Step 6. ____
   Step 7. Move the current card to another pile called Pile 2 and repeat from step 3
   Step 8. Maintain a variable **average** and initialize it to 0
   Step 9. Calculate average; divide **sum** by **count** and store the result in **average**

a. *<u>Step 5.</u>* Add *bill number* to **count**
   <u>Step 6.</u> Store *Total bill amount* in variable **sum**

b. *<u>Step 5.</u>* Increment **count**
   <u>Step 6.</u> Store *Total bill amount* in variable **sum**

c. *<u>Step 5.</u>* Add *bill number* to **count**
   <u>Step 6.</u> Add *Total bill amount* to variable **sum**

d. *<u>Step 5.</u>* Increment **count**
   <u>Step 6.</u> Add *Total bill amount* to variable **sum**

e. None of the above

4. What will the given procedure compute and store in variable **X** if executed on the "Shopping Bills" dataset?

<u>Step 1.</u> Arrange all cards in a single pile called Pile 1
<u>Step 2.</u> Maintain three variables **A**, **B**, **C** and initialize them to 0
<u>Step 3.</u> If Pile 1 is empty then stop the iteration and start from step 9
<u>Step 4.</u> Read the top card in Pile 1
<u>Step 5.</u> If the *shop name* is "SV Stores" and *customer name* is "Neeraja" then set **A** equal to 1
<u>Step 6.</u> If the *shop name* is "Big Bazaar" and *customer name* is "Neeraja" then set **B** equal to 1
<u>Step 7.</u> If the *shop name* is "Sun General" and *customer name* is "Neeraja" then set **C** equal to 1
<u>Step 8.</u> Move the current card to another pile called Pile 2 and repeat from step 3
<u>Step 9.</u> Maintain a variable **X** and initialize it to 0
<u>Step 10.</u> If **A** is equal to 1 then increment **X**
<u>Step 11.</u> If **B** is equal to 1 then increment **X**
<u>Step 12.</u> If **C** is equal to 1 then increment **X**

a. Numbers of times Neeraja has visited SV Stores
b. Numbers of times Neeraja has visited Sun General
c. Numbers of times Neeraja has visited Big Bazaar
d. Number of stores which Neeraja visited
e. Number of stores which Neeraja never visited
f. None of the above

5. The following information represents a card from the "Paragraph Words" dataset. During data entry, the clerk (or operator) might have made a mistake in one or more lines. Identify such lines with respect to the sanity of data. It is a Multiple Select Question (MSQ).

Line 1. Card number: 15
Line 2. Word: unplesant
Line 3. Word type: adjective
Line 4. Letter count: 0

a. Line 1
b. Line 2
c. Line 3
d. Line 4

Week 2: Practice Assignment

Note: Different font styles or punctuation symbols are used as follows

I. Italic + Underline: Steps of the procedure
II. Bold: Variables
III. Italic: A field from dataset
IV. " ": A value from dataset or dataset name

1. What will be the value of **X** after execution of the following procedure using the "Scores" dataset?

   Step 1. Arrange all cards in a single pile called Pile 1
   Step 2. Maintain a variable **X** and initialize it to 0
   Step 3. If Pile 1 is empty then stop the execution
   Step 4. Read the top card in Pile 1
   Step 5. If *Town/City* is "Chennai" and *TOTAL marks* > **X** then store *TOTAL marks* in **X**
   Step 6. Move the current card to another pile called Pile 2 and repeat from step 3

   Answer: 254 (Numerical input)

2. What will the values of **A** and **B** represent at the end of execution of the given procedure on the "Scores" dataset?

   Step 1. Arrange all cards in a single pile called Pile 1
   Step 2. Maintain two variables **A**, **B** and initialize them to 0
   Step 3. If Pile 1 is empty then stop the execution
   Step 4. Read the top card in Pile 1
   Step 5. If **A** < *Chemistry marks* then store *Chemistry marks* in **A**
   Step 6. If *Mathematics marks* < **B** then store *Mathematics marks* in **B**
   Step 7. Move the current card to another pile called Pile 2 and repeat from step 3

   a. Lowest marks of Chemistry and Highest marks of Mathematics respectively
   b. Highest marks of Chemistry and Lowest marks of Mathematics respectively
   c. Highest marks of Chemistry and Highest marks of Mathematics respectively
   d. Lowest marks of Chemistry and Lowest marks of Mathematics respectively
   e. None of the above

3. The following procedure finds the *Date of Birth* of the Physics topper student from the "Scores" dataset. But the programmer may have made mistakes in one or more steps. Identify all such steps (if any). It is a Multiple Select Question (MSQ).

   Step 1. Arrange all cards in a single pile called Pile 1
   Step 2. Maintain a variable **max** and initialize it to 0
   Step 3. Maintain a variable **DOB** and initialize it to "None"
   Step 4. If Pile 1 is empty then stop the execution
   Step 5. Read the top card in Pile 1
   Step 6. If *Physics marks* > **max** then store *Physics marks* in **max** and *Date of Birth* in **DOB**
   Step 7. Move the current card to another pile called Pile 2 and repeat from step 4

   a. Step 1
   b. Step 2
   c. Step 3
   d. Step 4
   e. Step 5
   f. Step 6

g. Step 7

h. No mistake

4. Answer the following sub-questions based on execution of the following pseudocode using the “Paragraph Words” dataset?

```
A = 0
B = 0
while (Pile 1 has more cards) {
        Read the top card X in Pile 1
        if (X.PartOfSpeech == “Noun”) {
                if (X.LetterCount > 6) {
                        A = A + 1
                }
                else {
                        B = B + 1
                }
        }
        Move X to Pile 2
}
```

i. What will the value of **A** represent at the end of execution of the above pseudocode?

a. Total number of nouns having letter count greater than 6

b. Total number of words having letter count greater than 6

c. Total number of nouns in the dataset

d. Total number of nouns having letter count greater than or equal to 5

e. None of the above

ii. What will be the value of **B** after execution of the above pseudocode?

Answer: 10 (Numerical input)

Practice Assignment

1. The following pseudocode is executed using the "Shopping bills" dataset. At the end of the execution, the variable **CountBBBA** captures the number of bills from Big Bazaar with total bill amount less than average total bill amount.Choose the correct code block to complete the pseudocode.

```
SumT = 0
Count = 0
while (Pile 1 has more cards) {
    Read the top card X in Pile 1
    SumT = SumT + X.TotalBillAmount
    Count = Count + 1
    Move X to Pile 2
}
AvgT = SumT / Count
CountBBBA = 0
while (Pile 2 has more cards) {
    Read the top card X in Pile 2
    ********************
    *    Fill the code    *
    ********************
    Move X to Pile 1
}
```

a.
```
if (X.TotalBillAmount < AvgT) {
    CountBBBA = CountBBBA + 1
}
```

b.
```
if (X.ShopName == "Big Bazaar" or X.TotalBillAmount < AvgT) {
    CountBBBA = CountBBBA + 1
}
```

c.
```
if (X.ShopName == "Big Bazaar" and X.TotalBillAmount < AvgT) {
    Count = Count + 1
}
```

d.
```
if (X.ShopName == "Big Bazaar" and X.TotalBillAmount < AvgT) {
    CountBBBA = CountBBBA + 1
}
```

2. The following pseudocode is executed using the “Scores” table to compute the number of students in each grade where grades are assigned based on Physics marks. Choose the correct code block to complete the pseudocode.

```
MinP = 100, MaxP = 0
while (Table 1 has more rows) {
    Read the first row X in Table 1
    if (X.Physics > MaxP) {
        MaxP = X.Physics
    }
    if (X.Physics < MinP) {
        MinP = X.Physics
    }
    Move X to Table 2
}
Interval = (MaxP - MinP) / 3
Mid2 = MinP + Interval
Mid1 = Mid2 + Interval
CountA = 0, CountB = 0, CountC = 0
while (Table 2 has more rows) {
    Read the first row X in Table 2
    if (X.Physics ≥ Mid1) {
        CountA = CountA + 1
    }
    ********************
    *    Fill the code    *
    ********************
    if (X.Physics < Mid2) {
        CountC = CountC + 1
    }
}
```

a.
```
if (X.Physics < Mid1 and X.Physics ≥ Mid2) {
    CountB = CountB + 1
}
```

b.
```
if (X.Physics ≥ Mid2) {
    CountB = CountB + 1
}
```

c.
```
if (X.Physics < Mid1 and X.Physics ≥ Mid2) {
    CountB = 1
}
```

d.
```
if (X.Physics ≤ Mid1 and X.Physics ≥ Mid2) {
    CountB = CountB + 1
}
```

3. The following pseudocode is executed using the “Scores” dataset. What will the values of **A** and **B** represent at the end of the execution?

```
AM = DoSomething("M")
AF = DoSomething("F")
A = 0, B = 0
while (Pile 2 has more cards) {
    Read the top card X in Pile 2
    if (X.Gender == "M" and X.Total < AF) {
        A = A + 1
    }
    if (X.Gender == "F" and X.Total < AM) {
        B = B + 1
    }
    Move X to Pile 1
}

Procedure DoSomething (Gen)
    Arrange all cards on Pile 1
    Sum = 0, Count = 0
    while (Pile 1 has more cards) {
        Read the top card X from Pile 1
        if (X.Gender == Gen) {
            Sum = Sum + X.Total
            Count = Count + 1
        }
        Move X to Pile 2
    }
    C = Sum / Count
    return (C)
End DoSomething
```

a. **A** = Number of male students with total marks less than the average total marks
   B = Number of female students with total marks less than the average total marks

b. **A** = Number of male students with total marks less than the average total marks of male students
   B = Number of female students with total marks less than the average total marks of female students

c. **A** = Number of female students with total marks less than the average total marks
   B = Number of male students with total marks less than the average total marks

d. **A** = Number of male students with total marks less than the average total marks of female students
   B = Number of female students with total marks less than the average total marks of male students

4. The following pseudocode is executed using the “Shopping bills” dataset. What will be the values of **Test1** and **Test2** at the end of the execution?

```
Let AvgT be the average total bill amount across all shops
CountSV = 0, CountBB = 0, CountSG = 0
while (Pile 1 has more cards) {
    Read the top card X from Pile 1
    if (X.TotalBillAmount < AvgT) {
        if (X.ShopName == “SV Stores”) {
            CountSV = CountSV + 1
        }
        if (X.ShopName == “Big Bazaar”) {
            CountBB = CountBB + 1
        }
        if (X.ShopName == “Sun Generals”) {
            CountSG = CountSG + 1
        }
    }
    Move X to Pile 2
}
Test1 = 0, Test2 = None
if (CountSV == 0 or CountBB == 0 or CountSG == 0) {
    Test1 = Test1 + 1
}
if (CountSV < CountSG) {
    if (CountBB < CountSV) {
        Test2 = “Big Bazaar”
    }
    else {
        Test2 = “SV Stores”
    }
}
else {
    if (CountBB < CountSG) {
        Test2 = “Big Bazaar”
    }
    else {
        Test2 = “Sun Generals”
    }
}
```

Answer: **Test1** = 0, **Test2** = “Big Bazaar” (Numerical and text input)

5. The following pseudocode is executed using the “Words” table. Let **A** and **B** be the variables that hold the values of letter counts of shortest and longest word, respectively. What will the value of **Count** represents at the end of the execution?

```
Count = 0
while (Table 1 has more cards) {
    Read the first row X from Table 1
    Move X to T2
    if (X.Word ends with a full stop) {
        Count = Count + DoSomething (T2, A, B)
        Clear all rows in T2
    }
}

Procedure DoSomething (Table 2, X, Y)
    C = 0, D = 0
    while (Table 2 has more rows) {
        Read the first row Z from Table 2
        if (Z.LetterCount ≤ X) {
            C = 1
        }
        if (Z.LetterCount ≥ Y) {
            D = 1
        }
        Move X to Table 3
    }
    if (C + D == 2) {
        return (1)
    }
    else {
        return (0)
    }
End DoSomething
```

a. Number of sentences with two shortest word or two longest word

b. Number of sentences with two shortest word and two longest word

c. Number of sentences with at least one shortest word or at least one longest word

d. Number of sentences with at least one shortest word and at least one longest word

e. None of the above

1. The given pseudocode is executed using the “Scores” table. What will the value of **Count** repersent at the end of the execution?

```
Count = 0
while (Table 1 has more rows) {
    Read the first row X in Table 1
    Move X to Table 2
    while (Table 1 has more rows) {
        Read the first row Y in Table 1
        Count = Count + DoSomething(X, Y)
        Move Y to Table 3
    }
    Move all rows from Table 3 to Table 1
}

Procedure DoSomething(A, B)
    if (A.Gender == B.Gender and A.CityTown == B.CityTown) {
        return (1)
    }
    else {
        return (0)
    }
End DoSomething
```

a. Number of pairs of students with the same gender

b. Number of pairs of students with the same city/town

c. Number of pairs of students with the same gender and the same City/Town

d. Number of pairs of students with the same gender or the same City/Town

2. Consider the pseudocode given in Question 1. The procedure **DoSomething** is rewritten such that the pseudocode computes the number of pairs of students with the same marks in at least one subject. Choose the correct choice to complete the pseudocode.

```
Procedure DoSomething(A, B)
    *********************
    *    Fill the code    *
    *********************
End DoSomething
```

a.
```
if (A.$Physics$ == B.$Physics$ and A.$Chemistry$ == B.$Chemistry$
  and A.$Mathematics$ == B.$Mathematics$) {
    return (1)
}
else {
    return (0)
}
```

b.
```
if (A.$Physics$ == B.$Physics$ or A.$Chemistry$ == B.$Chemistry$
  or A.$Mathematics$ == B.$Mathematics$) {
    return (1)
}
```

c.
```
if (A.$Physics$ == B.$Physics$ or A.$Chemistry$ == B.$Chemistry$
  or A.$Mathematics$ == B.$Mathematics$) {
    return (1)
}
else {
    return (0)
}
```

d.
```
if (A.$Physics$ == B.$Physics$ and A.$Chemistry$ == B.$Chemistry$
  and A.$Mathematics$ == B.$Mathematics$) {
    return (1)
}
```

3. The given pseudocode is executed using the “Words” table to count the number of pairs of words where both have the same part of speech and letter count. But the pseudocode have some mistkes. Identify all such mistakes (if any). It is a Multiple Select Question (MSQ).

```
1    Count = 0
3    while (Table 1 has more rows) {
4        Read the first row X in Table 1
5        Move X to Table 2
6        while (Table 1 has more rows) {
7            Read the first row Y in Table 1
8            Count = Count + DoTheyMatch(X, Y)
9            Move Y to Table 3
10       }
11       Move all rows from Table 3 to Table 2
12   }

13   Procedure DoTheyMatch(A, B)
14       if (A.PartOfSpeech == B.PartOfSpeech or A.LetterCount == B.LetterCount) {
15           return (1)
16       }
17       else {
18           return (0)
19       }
20   End DoTheyMatch
```

a. Error in Line 1

b. Error in Line 7

c. [✓ Error in Line 11]

d. Error in Line 13

e. [✓ Error in Line 14]

f. Error in Line 18

4. The following psedocode is executed using the “Scores” table. Assume that the procedure **GetCountOfMatched** will take a table as a parameter, and it will return the number of pairs of students in the input table where both have the same date of birth. Then, what will the value of **Count** represent at the end of the execution?

```
Count = 0
while (Table 1 has more rows) {
    Read the first row X in Table 1
    if ("JAN 01" ≤ X.DoB ≤ "MAR 31"){
        Move X to T1
    }
    if ("APR 01" ≤ X.DoB ≤ "JUN 30"){
        Move X to T2
    }
    if ("JUL 01" ≤ X.DoB ≤ "SEP 30"){
        Move X to T3
    }
    if ("OCT 01" ≤ X.DoB ≤ "DEC 31"){
        Move X to T4
    }
}
Count = GetCountOfMatched(T1)
Count = Count + GetCountOfMatched(T2)
Count = Count + GetCountOfMatched(T3)
Count = Count + GetCountOfMatched(T4)
```

a. Number of pairs of students with the same date of birth

b. Pairs of students with the same date of birth

c. Number of pairs of students with the same date of birth where the month of birth is on or before MARCH

d. Number of pairs of students in each table

5. What does the given pseudocode compute using the “Words” table?

```
Count = 0
while (Table 1 has more rows) {
    Read the first row X in Table 1
    if ($0 \leq$ DoSomething(X) $\leq 1$){
        Move X to T1
    }
    if ($2 \leq$ DoSomething(X) $\leq 3$){
        Move X to T2
    }
    if ($4 \leq$ DoSomething(X) $\leq 5$){
        Move X to T3
    }
    if (DoSomething(X) $\geq 6$){
        Move X to T4
    }
}

Procedure DoSomething (Y)
    i = 1
    Count = 0
    while (i $\leq$ Y.LetterCount) {
        if ($i^{th}$ letter of Y.Word is a vowel) {
            Count = Count + 1
        }
        i = i + 1
    }
    return (Count)
End DoSomething
```

a. Bins the rows in the “Words” table into four tables based on the letter count

b. Bins the rows in the “Words” table into four tables based on the vowel count of words

c. Bins the rows in the “Words” table into four tables based on the letter count and the vowel count

d. Bins the rows in the “Words” table into four tables based on the letter count or the vowel count

6. The following pseudocode is executed using the “Words” table. Consider the tables T1, T2, T3 and T4 computed in Question 5. Also, consider the procedure **DoSomething** in Question 5. Then, what will the value of **Count** represent at the end of the execution?

```
Count = GetCountOfMatched(T1)
Count = Count + GetCountOfMatched(T2)
Count = Count + GetCountOfMatched(T3)
Count = Count + GetCountOfMatched(T4)

Procedure GetCountOfMatched (Table 1)
    CountA = 0, i = 0
    while (Table 1 has more rows) {
        Read the first row X from Table 1
        Move X to Table 2
        while (Table 1 has more rows) {
            Read the first row Y from Table 1
            CountA = CountA + DoTheyMatch(X, Y)
            Move Y to Table 3
        }
        Move all rows from Table 3 to Table 1
    }
    return (CountA)
End GetCountOfMatched

Procedure DoTheyMatch (A, B)
    if (A.LetterCount == B.LetterCount
       and DoSomething(A) == DoSomething(B)) {
        return (1)
    }
    return (0)
End DoTheyMatch
```

a. Number of pairs of words with the same letter count

b. Number of pairs of words with the same letter count as well as vowel count

c. Number of pairs of words with the same letter count or the same vowel count

d. Number of pairs of words with the same letter count but the different vowel count

7. Let **X** and **Y** be two rows from the "Scores" table. Let **isFairPair**(**X**, **Y**) be a procedure to find whether **X** and **Y** is fair pair. Based on the procedure **isFairPair**, choose the correct option for the definition of fairness.

```
Procedure isFairPair (X, Y)
    XCount = 0, YCount = 0
    if (X.Physics > Y.Physics) {
        XCount = XCount + 1
    }
    if (X.Physics < Y.Physics) {
        YCount = YCount + 1
    }
    if (X.Chemistry > Y.Chemistry) {
        XCount = XCount + 1
    }
    if (X.Chemistry < Y.Chemistry) {
        YCount = YCount + 1
    }
    if (X.Mathematics > Y.Mathematics) {
        XCount = XCount + 1
    }
    if (X.Mathematics < Y.Mathematics) {
        YCount = YCount + 1
    }
    if (XCount > 0 and YCount > 0) {
        return (1)
    }
    else {
        return (0)
    }
End isFairPair
```

a. **X** and **Y** is said to be a *fair pair* if and only if **X** has more marks than **Y** in Physics and **Y** has more marks than **X** in Mathematics.

b. **X** and **Y** is said to be a *fair pair* if and only if **X** has more marks than **Y** in Mathematics and **Y** has more marks than **X** in Physics.

c. **X** and **Y** is said to be a *fair pair* if and only if **X** has more marks than **Y** in at least one subject and **Y** has more marks than **X** in at least one subject.

d. **X** and **Y** ise said to be a *fair pair* if and only if **X** has more marks than **Y** in one subject, **Y** has more marks than **X** in one subject and both got the same marks in one subject.

8. Let **X** and **Y** be two words from the "Words" table. The given procedure **CountPair** counts the number of sentences that contain both words **X** and **Y**. But the procedure may have mistakes. Identify all such mistakes (if any). It is a Multiple Select Question (MSQ).

```
Procedure CountPair (X, Y)
    XCount = 0, YCount = 0, Count = 0
    while (Table 1 has more rows) {
        Read the first row Z in Table 1
        if (Z.Word == X) {
            XCount = XCount + 1
        }
        if (Z.Word == Y) {
            YCount = YCount + 1
        }
        if (Z.Word ends with a full stop) {
            if (XCount > 0 or YCount > 0) {
                Count = Count + 1
            }
        }
        Move Z to Table 2
    }
    return (Count)
End CountPair
```

a. Condition to increment **XCount** is incorrect

b. Condition to increment **YCount** is incorrect

c. Condition to increment **Count** is incorrect

d. **XCount** and **YCount** are not reinitialized after reading the end of a sentence

BSCCS1001: Week-5, Practice Assignment

1. The following pseudocode is executed on the "Scores" table. What does the value **A** represent at the end of the execution? It is a Multiple Select Questions (MSQ).

```
S = [ ]
while (Table 1 has more rows) {
      Read the first row X in Table 1
      if (X.CityTown == "Madurai" or X.Gender == "F") {
            S = S ++ [X.SeqNo]
      }
      Move X to Table 2
}
A = length(S)
```

☐ Number of female students from Madurai

☐ (Number of female students) + (Number of students from Madurai)

☐ (Number of female students) + (Number of students from Madurai) - (Number of female students from Madurai)

☐ (Number of female students) + (Number of male students from Madurai)

☐ (Number of female students not from Madurai) + (Number of students from Madurai)

2. **topMath** and **topPhy** are lists of cards having Mathematics marks and Physics marks greater than 80 respectively. Each element in both lists is a record. The fields in the record are the same as the ones in the header of the "Scores" table. What will the value of the list **someList** represent at the end of the execution of the pseudocode given below?

```
someList = [ ]
foreach X in topMath {
    foreach Y in topPhy {
        if (X.SeqNo == Y.SeqNo and X.Chemistry > 80) {
            someList = someList ++ [X.Name]
        }
    }
}
```

- ◯ It stores the names of students who have scored above 80 in Chemistry.
- ◯ It stores the names of students who have scored above 80 in both Mathematics and Physics.
- ◯ It stores the names of students who have scored above 80 in at least one subject.
- ◯ It stores the names of students who have scored above 80 in all three subjects.

3. We call two sentences *similar* if both of them have the same number of words and satisfy the following condition:

   The i^{th} word in the first sentence has the same part of speech as the i^{th} word in the second sentence, for $1 \leq i \leq L$, where L is the total number of words in either sentence.

   aList and **bList** are lists that contain the part of speech of words in two sentences **A** and **B** respectively. **isSimilar** is a procedure that accepts these two lists as parameters and checks for the similarity of **A** and **B**. Select the correct code fragment to complete the pseudocode.

```
Procedure isSimilar(aList, bList)
    if (length(aList) ≠ length(bList)) {
        return (False)
    }
    cList = bList
    *************************
    *       Fill the code       *
    *************************
End isSimilar
```

○
```
foreach x in aList {
    if (x == first(cList)) {
        return (True)
    }
    cList = init(cList)
}
```

○
```
foreach x in aList {
    if (x ≠ first(cList)) {
        return (False)
    }
    cList = init(cList)
}
```

```
○ foreach x in aList {
      if (x ≠ first(cList)) {
          return (False)
      }
      cList = rest(cList)
  }
```

```
○ foreach x in aList {
      if (x ≠ first(cList)) {
          return (False)
      }
      cList = rest(cList)
  }
  return (False)
```

```
○ foreach x in aList {
      if (x ≠ first(cList)) {
          return (False)
      }
      cList = rest(cList)
  }
  return (True)
```

4. For a given train, we have a list called **days** that contains the sequence of entries contained in the column *Day* in the “Trains” dataset. As an example, for the train “12281”, **days** will be the following list of integers: [1, 1, 1, 1, 2, 2].

   Madhuram is a passenger who boards this train at its starting station and gets down at its ending station. He has a peculiar habit of treating himself to exactly one sweet at midnight whenever he is on a train. Which of the following expressions gives the number of “midnight sweets” that he consumes during the course of his journey? It is a Multiple Select Question (MSQ).

   - ☐ **last**(**days**) - **first**(**days**)
   - ☐ **last**(**days**) - **first**(**days**) + 1
   - ☐ **last**(**init**(**days**)) - **first**(**rest**(**days**))
   - ☐ **length**(**days**)
   - ☐ **last**(**days**) - 1
   - ☐ **length**(**days**) - 1

5. **trainList** is a non-empty list that contains information about all trains associated with a station. Each element in **trainList** is a list of running days for a train. For example, given a station that handles two trains, we could have [[“M”, “W”], [“Sa”]]. Assume that all the elements of **trainList** are non-empty lists. Consider the following pseudocode.

```
m = 0, tu = 0, w = 0, th = 0, fr = 0, sa = 0, su = 0
foreach x in trainList {
    foreach y in x {
        if (y == “M”) {
            m = 1
        }
        if (y == “Tu”) {
            tu = 1
        }
        if (y == “W”) {
            w = 1
        }
        if (y == “Th”) {
            th = 1
        }
        if (y == “F”) {
            f = 1
        }
        if (y == “Sa”) {
            sa = 1
        }
        if (y == “Su”) {
            su = 1
        }
    }
}
```

The above pseudocode is executed on the **trainList** for some arbitrary station. After execution, we run the following code.

```
someVar = 0
someVar = 7 - (m + tu + w + th + f + sa + su)
```

(a) What does the variable **someVar** store at the end of execution?

- ◯ It stores the number of days in a week when at least one train visits the station.
- ◯ It stores the number of days in a week when no train visits the station.
- ◯ It stores the number of trains that visit the station in a week.
- ◯ It does not store anything meaningful.

(b) Which of the following is an accurate bound for the variable **someVar**?

- ◯ $0 \leq \textbf{someVar} \leq 7$
- ◯ $0 < \textbf{someVar} \leq 7$
- ◯ $0 \leq \textbf{someVar} < 7$
- ◯ $0 < \textbf{someVar} < 7$

Computational Thinking

Week - 6

Practice Assignment

1. The following pseudocode is executing using the "Scores" table. What will the value of **Z** represent at the end of the execution?

[3 marks]

```
D = { }
A = 0, C = 0
while (Table 1 has more rows){
    Read the first row X in Table 1
    A = A + X.Total
    C = C + 1
    if (isKey(D, X.CityTown)) {
        D[X.CityTown]["Total"] = D[X.CityTown]["Total"] + X.Total
        D[X.CityTown]["Count"] = D[X.CityTown]["Count"] + 1
    }
    else {
        D[X.CityTown] = {"Total": X.Total, "Count": 1 }
    }
    Move X to Table 2
}
Avg = A / C
Z = 0
foreach B in keys(D) {
    D[B]["Average"] = D[B]["Total"] / D[B]["Count"]
    if (D[B]["Average"] > Avg) {
        Z = Z + D[B]["Count"]
    }
}
```

a. Number of students from the cities with the average total marks of the city is more than the average total marks of the dataset

b. Number of students from the cities with the average total marks of the city is less than the average total marks of the dataset

c. Number of cities with the average total marks less than the average total marks of the dataset

d. Number of cities with the average total marks more than the average total marks of the dataset

2. The following pseudocode is executed using the “Shopping Bills” dataset. The variable **D** is a dictionary that maintains the following information: the category of the item purchased by a customer. Choose the correct code fragment to complete the pseudocode. [5 Marks]

```
D = { }
while (Pile 1 has more cards) {
    Read the top card X in Pile 1
    ********************
    *    Fill the code    *
    ********************
    Move X to Pile 2
}
```

a.
```
foreach A in X.ItemList {
    D[X.CustomerName][A] = True
}
```

b.
```
if (isKey(D, X.CustomerName)) {
    D[X.CustomerName] = { }
}
foreach A in X.ItemList {
    D[X.CustomerName][A.Category] = True
}
```

c.
```
if (not isKey(D, X.CustomerName)) {
    D[X.CustomerName] = { }
}
else {
    foreach A in X.ItemList {
        D[X.CustomerName][A] = True
    }
}
```

d.
```
if (not isKey(D, X.CustomerName)) {
    D[X.CustomerName] = { }
}
foreach A in X.ItemList {
    D[X.CustomerName][A.Category] = True
}
```

3. What does the following pseudocode compute when it is executed using the “Shopping bills” dataset. [4 Marks]

```
BB = { }
while (Pile 1 has more cards) {
    Read the top card X in Pile 1
    if (X.ShopName == “Big Bazaar”) {
        BB = updateDictionary(BB, X)
    }
    Move X to Pile 2
}
A = getMaxKeyByKey(BB)

Procedure updateDictionary(D, Y)
    foreach A in Y.ItemList {
        if (isKey(D, A.Category)) {
            D[A.Category] = D[A.Category] + 1
        }
        else{
            D[A.Category] = 1
        }
    }
    return (D)
End updateDictionary

Procedure getMaxKeyByKey(D)
    A = “None”, B = 0
    foreach Y in keys(D) {
        if (B < D[Y]) {
            A = Y
            B = D[Y]
        }
    }
    return (A)
End getMaxKeyByKey
```

a. Finds the item category from Big Bazaar
b. Finds the top item category by item category count from Big Bazaar
c. Finds the most frequent item category from Big Bazaar
d. Finds the top item category by item quantity from Big Bazaar

4. The following pseudocode is executed using the "`station_wise`" cards of the "Train" dataset. What will the value of **C** represent at the end of the execution? [3 Marks]

```
D = { "M": 0, "Tu": 0, "W": 0, "Th": 0, "F": 0, "Sa": 0, "Su": 0}
while (Pile 1 has more cards) {
    Read the top card X in Pile 1
    if (X.StationName == "Mumbai") {
        foreach A in X.TrainList {
            foreach B in A.Days {
                D[B] = D[B] + 1
            }
        }
    }
    Move X to Pile 2
}
C = [ ], Y = 0
foreach B in keys(D) {
    if (Y == D[B]) {
        C = C ++ [B]
    }
    if (Y < D[B]) {
        C = [B]
        Y = D[B]
    }
}
```

a. The busiest day of the Mumbai station

b. List of busiest days of the Mumbai station

c. The least busy day of the Mumbai station

d. List of least busy days of the Mumbai station

5. The following pseudocode is executed using the "Words" table. At the end of the execution, **C** captures the number of words with either all letters are distinct, or at least three letters are appeared at least twice. Choose the correct code-fragment to complete the pseudocode. [5 Marks]

```
C = 0
while (Table 1 has more rows) {
    Read the first row X in Table 1
    i = 1, D = { }
    while (i ≤ X.LetterCount) {
        B = i^th letter in X.Word
        if (isKey(D, B)) {
            D[B] = D[B] + 1
        }
        else{
            D[B] = 1
        }
        i = i + 1
    }
    ********************
    *    Fill the code    *
    ********************
    Move X to Table 2
}
```

a.
```
Found = True, A = 0
foreach Y in keys(D) {
    if (D[Y] > 1) {
        A = A + 1
    }
    else{
        Found = False
    }
}
if (Found or A ≥ 3) {
    C = C + 1
}
```

b.

```
Found = True, A = 0
foreach Y in keys(D) {
    if (D[Y] > 1) {
        Found = False
        A = A + 1
    }
}
if (Found or A ≥ 3) {
    C = C + 1
}
```

c.

```
Found = True, A = 0
foreach Y in keys(D) {
    if (D[Y] > 1) {
        Found = False
        A = A + 1
    }
    else{
        Found = True
    }
}
if (Found or A ≥ 3) {
    C = C + 1
}
```

d.

```
Found = True, A = 0
foreach Y in keys(D) {
    if (D[Y] > 1) {
        Found = False
    }
    else{
        A = A + 1
    }
}
if (Found or A ≥ 3) {
    C = C + 1
}
```

BSCCS1001: Week-7, Practice Assignment

1. Consider the following graph generated from the "Scores" table. Each node in this graph corresponds to a student from the table. There is an edge between two nodes if the difference in the physics marks of the corresponding students is at most five.

   The graph is represented by the matrix **A**. *SeqNo* is used to label the nodes in the graph. The following pseudocode computes the matrix **A**. Select the correct code fragment to complete it.

```
marks = readPhysicsMarks()
n = length(keys(marks))

********************
*      Fill the code      *
********************


Procedure readPhysicsMarks()
    marks = { }
    while (Table 1 has more rows) {
        Read the first row X in Table 1
        marks[X.SeqNo] = X.Physics
        Move X to Table 2
    }
    return (marks)
End readPhysicsMarks
```

○
```
A = createMatrix(n, n)
foreach i in keys(marks) {
    foreach j in keys(marks) {
        if (-5 ≤ marks[i] - marks[j] ≤ 5) {
            A[i][j] = 1
        }
    }
}
```

○
```
A = createMatrix(n, n)
foreach i in keys(marks) {
    foreach j in keys(marks) {
        if (-5 < marks[i] - marks[j] < 5) {
            A[i][j] = 1
        }
    }
}
```

○
```
foreach i in keys(marks) {
    foreach j in keys(marks) {
        if (-5 < marks[i] - marks[j] < 5) {
            A[i][j] = 1
        }
    }
}
```

○
```
A = createMatrix(n, n)
foreach i in keys(marks) {
    foreach j in keys(marks) {
        if (0 ≤ marks[i] - marks[j] ≤ 5) {
            A[i][j] = 1
        }
    }
}
```

2. Consider the following table containing the scores of n students in a class. Assume that all scores are between 0 and 100. The scores in the first and last row are shown as an example. The actual scores may be different. So, refer only to the table's header to answer all questions:

S.No	**Score**
0	50
$\vdots$	$\vdots$
$n-1$	89

This information is stored in a dictionary **S**, with serial numbers as keys and the corresponding scores as values. A graph ***G*** is generated from this table in the following manner:

Given a pair of students (i, j), there is an edge from i to j if and only if i's score is strictly greater than that of j. This graph is represented by a matrix **A**. Assume that the dictionary **S** has already been computed. Now, study the following pseudocode.

```
n = length(keys(S))
A = createMatrix(n, n)
foreach r in rows(A) {
    foreach c in columns(A) {
        if (S[r] > S[c]) {
            A[r][c] = 1
        }
    }
}
```

Answer the following questions **after** executing the pseudocode given above.

(a) A row r of the matrix has three entries that are equal to one and the rest are zero. Which of the following statements about the student r is true?

◯ There are exactly three students who have scored more than student r.

◯ There are at least three students who have scored less than student r.

◯ There are exactly three students who have scored less than student r.

◯ There are exactly three students who have scored less than or equal to student r.

(b) If r and c are two students with unequal scores, which of the following statements are true? It is a Multiple Select Question (MSQ).

- ☐ If $\mathbf{A}[\mathbf{r}][\mathbf{c}]$ is 1, then $\mathbf{A}[\mathbf{c}][\mathbf{r}]$ is also 1
- ☐ If $\mathbf{A}[\mathbf{r}][\mathbf{c}]$ is 1, then $\mathbf{A}[\mathbf{c}][\mathbf{r}]$ is 0
- ☐ If $\mathbf{A}[\mathbf{r}][\mathbf{c}]$ is 0, then $\mathbf{A}[\mathbf{c}][\mathbf{r}]$ is 1
- ☐ If $\mathbf{A}[\mathbf{r}][\mathbf{c}]$ is 1, then there is insufficient information to determine the value of $\mathbf{A}[\mathbf{c}][\mathbf{r}]$

(c) Which of the following statements about the matrix $\mathbf{A}$ are true? It is a Multiple Select Question.

- ☐ There is at least one row that has only ones in it.
- ☐ There is at least one row that has only zeros in it.
- ☐ There is at least one column that has only ones in it.
- ☐ There is at least one column that has only zeros in it.

3. Continuing with the previous question, assume that the matrix **A** has been computed. Study the following pseudocode.

```
n = length(keys(S))
k = 0
someScore = S[k]
walk = True
hops = 0
while (walk) {
    walk = False
    foreach c in columns(A) {
        if (A[k][c] == 1) {
            k = c
            someScore = S[k]
            walk = True
            hops = hops + 1
            exitloop
        }
    }
}
```

Answer the following four questions **after** executing the pseudocode given above. The values of all variables mentioned in the following questions represent the state of these variables at the end of the execution of the above pseudocode. All four questions given below are independent of each other.

(a) Which of the following statements about the variable **someScore** is true, irrespective of the contents of the score-table?

○ It is equal to the score of student 0.
○ It is equal to the score of student $n - 1$.
○ It is the minimum score in the table.
○ It is the maximum score in the table.

(b) The value of **hops** is equal to 0. Which of the following statements are true? It is a Multiple Select Question (MSQ).

☐ Student k has scored the maximum marks.
☐ Student k has scored the minimum marks.
☐ The value of k is equal to 0.
☐ Insufficient information to determine the value of k.

(c) The variable **hops** satisfies which of the following inequalities?

○ $0 \leq \textbf{hops} < \frac{n}{2}$
○ $0 \leq \textbf{hops} \leq \frac{n}{2}$
○ $0 \leq \textbf{hops} \leq n - 1$
○ $1 \leq \textbf{hops} \leq n$

(d) Which of the following statements are true after executing the pseudocode given below? It is a Multiple Select Question (MSQ).

```
sum = 0
foreach c in columns(A) {
    sum = sum + A[k][c]
}
```

☐ **sum** is equal to 0.
☐ **sum** is equal to $n - 1$.
☐ **sum** is the in-degree of node **k** in the graph.
☐ **sum** is the out-degree of node **k** in the graph.

4. Consider the graph $\boldsymbol{G}$ that we worked with in the previous two problems. $\boldsymbol{G_r}$ is a graph that is obtained in the following manner:

   Given a pair of students $(i,\ j)$, there is an edge from i to j if and only if i's score is strictly less than that of j.

   B is a matrix that represents $\boldsymbol{G_r}$.

   (a) The following pseudocode is used to construct the matrix **B** from **A**. Assume that **A** has already been computed. Select the correct code fragment to complete the pseudocode. It is a Multiple Select Question (MSQ).

```
n = length(keys(A))
B = createMatrix(n, n)
foreach i in rows(A) {
    foreach j in columns(A) {
        ********************
        *    Fill the code     *
        ********************
    }
}
```

   - ☐ **B**[**i**][**j**] = **A**[**j**][**i**]
   - ☐ **B**[**j**][**i**] = **A**[**i**][**j**]
   - ☐ if (**A**[**i**][**j**] == 0) {
         B[**i**][**j**] = 1
     }
   - ☐ if (**A**[**j**][**i**] == 1) {
         B[**i**][**j**] = 1
     }
   - ☐ **B**[**i**][**j**] = 1 - **A**[**i**][**j**]

   (b) For a pair of nodes (i, j), the following expression evaluates to true:
   A[**i**][**j**] == **B**[**i**][**j**]
   Based on the above observation, which of the following statements evaluates to true? **S** is the dictionary that represents the table in the previous question.

   - ◯ **S**[**i**] == **S**[**j**]
   - ◯ **S**[**i**] > **S**[**j**]
   - ◯ **S**[**i**] < **S**[**j**]
   - ◯ Insufficient information

BSCCS1001: Week-8 Pracice Assignment

Common data for all questions in this assignment

Consider the following condensed version of the "Trains" dataset. There are a total of n stations, with stations being indexed from 0 to $n-1$. There are M rows in the table. Each row contains information about a train that connects two stations without any stops in between.

SeqNo	**Train**	**Departure**	**Arrival**	**Distance**
0	12259	0	1	266
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
r	t	i	j	d
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$M-1$	12281	4	11	206

For example, the r^{th} row tells us that train t departs from station i and arrives at station j after covering a distance of d kilometers *without stopping* at any intermediate station. In other words, the r^{th} row in this table corresponds to two consecutive rows in the card for train t in the "Trains" dataset. Therefore, each train t will occupy multiple rows in this table.
This scenario is modeled as a graph and is represented by a matrix $\mathbf{A}$. Each node in the graph corresponds to a station. Assume that the value of n is already given to you. Consider the following pseudocode.

```
$\mathbf{S}$ = { }
while (Table 1 has more rows) {
    Read the first row $\mathbf{X}$ in Table 1
    $\mathbf{S}[\mathbf{X}.\mathit{SeqNo}]$ = { }
    $\mathbf{S}[\mathbf{X}.\mathit{SeqNo}]$[“train”] = $\mathbf{X}.\mathit{Train}$
    $\mathbf{S}[\mathbf{X}.\mathit{SeqNo}]$[“depart”] = $\mathbf{X}.\mathit{Departure}$
    $\mathbf{S}[\mathbf{X}.\mathit{SeqNo}]$[“arrive”] = $\mathbf{X}.\mathit{Arrival}$
    $\mathbf{S}[\mathbf{X}.\mathit{SeqNo}]$[“dist”] = $\mathbf{X}.\mathit{Distance}$
    Move $\mathbf{X}$ to Table 2
}


$\mathbf{A}$ = $\mathbf{createMatrix}(\mathbf{n}, \mathbf{n})$
foreach $\mathbf{r}$ in $\mathbf{rows}(\mathbf{A})$ {
    foreach $\mathbf{c}$ in $\mathbf{columns}(\mathbf{A})$ {
        $\mathbf{A}[\mathbf{r}][\mathbf{c}]$ = { }
    }
}


foreach $\mathbf{x}$ in $\mathbf{keys}(\mathbf{S})$ {
    $\mathbf{r}$ = $\mathbf{S}[\mathbf{x}]$[“depart”]
    $\mathbf{c}$ = $\mathbf{S}[\mathbf{x}]$[“arrive”]
    $\mathbf{t}$ = $\mathbf{S}[\mathbf{x}]$[“train”]
    $\mathbf{d}$ = $\mathbf{S}[\mathbf{x}]$[“dist”]
    $\mathbf{A}[\mathbf{r}][\mathbf{c}][\mathbf{t}] = \mathbf{d}$
}
```

1. If $(i,\ j)$ is a pair of stations, which of the following statements about the dictionary **A**[i][j] are true? It is a Multiple Select Question (MSQ).

   ☐ Each key corresponds to a train that goes from station i to j without stopping at any intermediate station.

   ☐ Each key corresponds to a train that goes from station i to j. It may stop at multiple stations between i and j.

   ☐ The value corresponding to key t of the dictionary is the distance between stations i and j on t's route.

   ☐ The value corresponding to key t of the dictionary is the minimum distance between stations i and j on t's route.

2. What does the value **length**(**keys**(**A**[**i**][**j**])) represent?

   ◯ It is the number of trains that go from station i to j that stop at at least one intermediate station.

   ◯ It is the number of trains that go from station i to j without stopping at any intermediate station.

   ◯ It is the number of trains that go from station j to i without stopping at any intermediate station.

   ◯ It is the number of trains that go from station i to j that stop at at most one intermediate station.

3. **ijMin** is a procedure that accepts a pair of stations (i, j), and the matrix **A** as input. It returns a train which goes from i to j by covering the least distance, without stopping at any intermediate station. If there is no train connecting these two stations, the procedure returns -1. The pseudocode may have mistakes. Identify all of them (if any). It is a Multiple Select Question (MSQ).

```
1    Procedure ijMin(i, j, A)
2        if (length(keys(A[i][j])) == 0) {
3            return (False)
4        }
5        train = first(keys(A[i][j]))
6        min = A[i][j][train]
7        foreach k in rows(A[i][j]) {
8            dist = A[i][j][k]
9            if (dist < min) {
10               min = dist
11               train = k
12           }
13       }
14       return (train)
15   End ijMin
```

☐ Error in line 3

☐ Error in line 5

☐ Error in line 6

☐ Error in line 7

☐ Error in line 8

☐ Error in line 9

☐ Error in line 10

☐ Error in line 11

☐ Error in line 14

4. Let (i, j) be a pair of stations. Which of the following statements are true after executing the pseudocode given below? It is a Multiple Select Question (MSQ).

```
iToj = False
if (length(keys(A[i][j])) ≠ 0) {
    iToj = True
}
jToi = False
if (length(keys(A[j][i])) ≠ 0) {
    jToi = True
}
iBothj = False
if (iToj and jToi)) {
    iBothj = True
}
```

- ☐ If **iToj** is True and **jToi** is False, then **iBothj** is True
- ☐ If **iToj** is True and **jToi** is True, then **iBothj** is True
- ☐ If **iBothj** is True, then **iToj** is True and **jToi** is True
- ☐ If **iBothj** is False, then **iToj** is False and **jToi** is False
- ☐ If **iBothj** is False, then **iToj** is False and **jToi** is True
- ☐ If **iBothj** is False, then **iToj** is True and **jToi** is False
- ☐ If **iBothj** is False, then **iToj** is False or **jToi** is False

5. **someProc** is a procedure that accepts three parameters: a train, a station and the matrix **A** that was computed in the previous question. Which of the following statements about its return value are true? It is a Multiple Select Question (MSQ).

```
Procedure someProc(train, stn, A)
    foreach r in rows(A) {
        if (isKey(A[r][stn], train)) {
            foreach c in columns(A) {
                if (isKey(A[stn][c], train)) {
                    return (False)
                }
            }
            return (True)
        }
    }
    return (False)
End someProc
```

- ☐ **someProc** returns True if **stn** is the starting station for **train**
- ☐ **someProc** returns False if **stn** is the starting station for **train**
- ☐ **someProc** returns True if **stn** is the ending station for **train**
- ☐ **someProc** returns False if **stn** is the ending station for **train**
- ☐ **someProc** returns True if **stn** is an intermediate station on **train**'s route
- ☐ **someProc** returns False if **stn** is an intermediate station on **train**'s route
- ☐ **someProc** returns True if **stn** is not on **train**'s route
- ☐ **someProc** returns False if **stn** is not on **train**'s route

BSCCS1001: Week-9, Practice Assignment

1. **uniq** is a procedure that accepts a list **L** as input. It returns the unique elements in the list after removing duplicates. Select the correct code fragment to complete the pseudocode.

```
Procedure uniq(L)
    if (length(L) <= 1) {
        return (L)
    }
    lastElem = last(L)
    initElems = init(L)
    ********************
    *    Fill the code    *
    ********************
End uniq
```

- ◯
```
if (member(initElems, lastElem)) {
    return (uniq(initElems))
}
else {
    return (uniq(initElems) ++ lastElem)
}
```

- ◯
```
if (member(initElems, lastElem)) {
    return (initElems)
}
else {
    return (initElems ++ [lastElem])
}
```

- ◯
```
if (member(initElems, lastElem)) {
    return (initElems)
}
else {
    return (uniq(initElems) ++ [lastElem])
}
```

○

```
if (member(initElems, lastElem)) {
    return (uniq(initElems))
}
else {
    return (uniq(initElems) ++ [lastElem])
}
```

2. A word is said to be a palindrome if the word obtained by reversing it is the same as the original word. For example, "madam" is a palindrome. The following pseudocode picks up a word from the "Words" table and checks if it is a palindrome or not. The result is stored in a boolean variable called **flag**. Select the correct implementation of the procedure **isPalindrome**. This procedure must return True if the word is a palindrome and False otherwise. It is a Multiple Select Question (MSQ).

```
Pick a row X from the "Words" Table
wordList = wordToList(X)
flag = isPalindrome(wordList)


********************
*      Fill the code      *
********************


Procedure wordToList(X)
    i = 1
    chars = [ ]
    while (i <= X.LetterCount) {
        chars = chars ++ [$i^{th}$ letter of X.Word]
        i = i + 1
    }
    return (chars)
End wordToList
```

☐ **Procedure isPalindrome(L)**

```
    if (length(L) <= 1) {
        return (True)
    }
    if (first(L) ≠ last(L)) {
        return (False)
    }
    else {
        return (isPalindrome(init(rest(L))))
    }
End isPalindrome
```

☐ **Procedure isPalindrome(L)**

```
    if (length(L) == 1) {
        return (True)
    }
    if (first(L) == last(L)) {
        return (isPalindrome(init(rest(L))))
    }
    else {
        return (False)
    }
End isPalindrome
```

☐ **Procedure isPalindrome(L)**

```
    if (length(L) <= 1) {
        return (True)
    }
    if (first(L) == last(L)) {
        return (isPalindrome(init(rest(L))))
    }
    else {
        return (False)
    }
End isPalindrome
```

☐ **Procedure isPalindrome(L)**

```
    if (length(L) <= 1) {
        return (True)
    }
    if (first(L) == last(L)) {
        return (True)
    }
    else {
        return (isPalindrome(init(rest(L))))
    }
End isPalindrome
```

Consider the following pseudocode. Assume that **a** and **b** are two positive integers.

```
Procedure calculate(a, b)
    if (a < b) {
        return (calculate(b, a))
    }
    if (a == b) {
        return (b)
    }
    diff = a − b
    if (diff > b) {
        return (calculate(diff, b))
    }
    else {
        return (calculate(b, diff))
    }
End calculate
```

3. What is the return value of **calculate**(**12**, **3**)?
   - ○ 0
   - ○ 1
   - ○ 3
   - ○ 4
   - ○ 12

4. What is the return value of **calculate**(14, 17)?
   - ○ 0
   - ○ 1
   - ○ 14
   - ○ 17

5. How many times is the procedure **calculate** called in order to compute **calculate**(14, 17)? Include the first call in your answer.

   - ○ 1
   - ○ 8
   - ○ 9
   - ○ 10

6. For any two positive integers **a** and **b**, what does **calculate**(**a**, **b**) return?

   - ○ It returns the LCM of the two numbers.
   - ○ It returns the HCF of the two numbers.
   - ○ It returns the LCM of the two numbers provided **a** > **b**.
   - ○ It returns the HCF of the two numbers provided **a** > **b**.

7. What is the result of the following expression? Assume that **a** and **b** are two positive integers.

   calculate(**a**, **b**) == **calculate**(**b**, **a**)

   - ○ True
   - ○ False
   - ○ It is True if and only if **a** is equal to **b**
   - ○ Insufficient information

Computational Thinking

Week - 10

Practice Assignment

1. Consider the definition of the object datatype **ClassTopper**. [(5+5) Marks]

```
ClassTopper
    private fields:
        tValue, markList
    private procedures:
        Procedure findTopMark()
            foreach A in markList {
                if (tValue < A) {
                    tValue = A
                }
            }
        End findTopMark
    public procedures:
        Procedure initialize()
            tValue = -1
            markList = [ ]
        End initialize

        Procedure addMark(mark)
            tValue = -1
            markList = markList ++ [mark]
        End addMark

        Procedure isTopper(mark)
            if (tValue == -1) {
                findTopMark()
            }
            if (tValue == mark) {
                return (True)
            }
            return (False)
        End isTopper
End ClassTopper
```

i) Let **ct** be an object of **ClassTopper**. Choose the correct statement(s) based on the definition of **ClassTopper**. It is a Multiple Select Question (MSQ).

a. **ct**.**findTopMark**() will return the maximum marks of the object **ct**.

b. The procedure **isTopper** will not work properly if the procedure **addMark** does not set **tValue** to -1.

c. **ct**.**tValue** and **ct**.**markList** can be updated directly.

d. **length**(**ct**.**markList**) gives the number of marks added to the **markList**.

ii) Let **MTop** and **PTop** be the **ClassTopper** objects. The following pseudocode is executed using the "Scores" table. At the end of the execution, **C** captures the number of students who are topper in both Physics and MAthematics. Choose the correct statement to fill if-condition in Line 12.

```
1   MTop.initialize()
2   PTop.initialize()
3   while (Table 1 has more rows) {
4       Read the first row X in Table 1
5       MTop.addMark(X.Mathematics)
6       PTop.addMark(X.Physics)
7       Move X to Table 2
8   }
9   C = 0
10  while (Table 2 has more rows) {
11      Read the first row X in Table 2
12      if (*** Fill the condition ***) {
13              C = C + 1
14          }
15      }
16      Move X to Table 1
17  }
```

a. **PTop.isTopper**(**X**.*Physics*)
b. **MTop.isTopper**(**X**.*Mathematics*)
c. **MTop.isTopper**(**X**.*Mathematics*) and **PTop.isTopper**(**X**.*Physics*)
d. None of the above.

2. Consider the definition of the object datatype **BankAccount**. [5 Marks]

```
BankAccount
    private fields:
        balance, minBalance
    private procedures:
        Procedure checkBalance(amount)
            if (balance - amount < minBalance) {
                return (False)
            }
            return (True)
        End checkBalance
    public procedures:
        Procedure initialize()
            balance = 100
            minBalance = 100
        End initialize

        Procedure deposit(amount)
            if (amount ≤ 0) {
                return (False)
            }
            balance = balance + amount
            return (True)
        End deposit

        Procedure withdraw(amount)
            if (checkBalance(amount)) {
                balance = balance - amount
                return (True)
            }
            else {
                return (False)
            }
        End withdraw
End BankAccount
```

Let **ba** be a **BankAccount** object. Choose the correct statement(s) about **ba**. It is a Multiple Select Question (MSQ).

a. **ba.checkBalance** cannot be updated directly.

b. **ba.balance** can be updated directly.

c. None of the procedures in **BankAccount** return the value of **balance**.

d. **deposit** and **withdraw** are the only procedures that update the object **ba**.

3. Consider the definition of the object datatype **Graph**. [5 Marks]

```
Graph
    private fields:
        matrix
    public procedures:
        Procedure initialize(n)
            matrix = createMatrix(n, n)
        End initialize

        Procedure isEdge(v, u)
            if (matrix[u][v] == 1) {
                return (True)
            }
            return (False)
        End isEdge

        Procedure addEdge(edge)
            u = first(edge)
            v = last(edge)
            matrix[u][v] = 1
        End addEdge
End Graph
```

Add a public procedure **removeEdge** to **Graph** as follows: For an object **G** of **Graph**, **G**.**removeEdge**(**edge**) removes **edge** from **G** if **edge** exists in **G** and returns True, otherwise, returns False. Choose the correct implementation of the procedure **removeEdge**.

a.

```
Procedure removeEdge(edge)
    u = first(edge)
    v = last(edge)
    if (not(isEdge(u, v))) {
        matrix[u][v] = 0
        return (True)
    }
    return (False)
End removeEdge
```

b.

```
Procedure removeEdge(edge)
    u = first(edge)
    v = last(edge)
    if (isEdge(u, v)) {
        matrix[u][v] = 0
        return (True)
    }
    return (False)
End removeEdge
```

c.

```
Procedure removeEdge(edge)
    u = first(edge)
    v = last(edge)
    if (not(isEdge(u, v))) {
        matrix[u][v] = 0
        return (False)
    }
    return (True)
End removeEdge
```

d.

```
Procedure removeEdge(edge)
    u = first(edge)
    v = last(edge)
    if (isEdge(u, v)) {
        matrix[u][v] = 0
        return (False)
    }
    return (True)
End removeEdge
```

BSCCS1001: Week-11 Practice Assignment

1. Let object datatype **Customer** be defined as follows:

```
Customer
    private fields:
        balance;
    public procedures:
        Procedure getBalance()
            return (balance);
        End getBalance
        Procedure setBalance(newBalance)
            balance = newBalance;
        End setBalance
End Customer
```

Let **cust1** and **cust2** be **Customer** objects. Consider the pseudocode for two bank transactions T_1 and T_2, given below, that transfer ₹100 from Customer 1 to Customer 2. T_1 debits ₹100 from the account of Customer 1 and T_2 credits ₹100 to the account of Customer 2. These transactions execute in parallel and the individual steps may be interleaved.

```
/* Transaction T1 */
L1: balanceCust1 = cust1.getBalance()                Reads balance of Cust 1
L2: balanceCust1 = balanceCust1 −100      Subtracts 100 from balanceCust1
L3: cust1.setBalance(balanceCust1)                      Saves the new balance
```

```
/* Transaction T2 */
P1: balanceCust2 = cust2.getBalance()                Reads balance of Cust 2
P2: balanceCust2 = balanceCust2 +100           Adds 100 to balanceCust2
P3: cust2.setBalance(balanceCust2)                      Saves the new balance
```

Assuming that Customer 1 and Customer 2 had a balance of ₹1000 and ₹2000, respectively, in their accounts, at what steps in the transactions are we likely to see amounts not matching the expected balances ₹900 and ₹2100 respectively?

- ◯ T_1 is at L2 and T_2 is at P2
- ◯ T_1 is at L1 and T_2 has completed P3
- ◯ T_1 has completed L3 and T_2 has completed P3
- ◯ T_1 has completed L3 and T_2 is at P1
- ◯ T_1 has completed L1 and T_2 is at P2

2. Two data entry clerks C_1 and C_2 are booking sleeper class tickets from Kolkata to New Delhi for passengers P_1 and P_2 respectively on train 12301. There is exactly one sleeper ticket left on 12301 and there is no provision to book RAC and/or waiting list tickets. The tables below list the steps of booking by clerks C_1 and C_2.

Time	**Clerk** C_1
t_1	read **FreeBerth**

t_2	if **FreeBerth** is not null,
	DataForBerth = Data of P_1
	else return;

t_3	save **DataForBerth** **FreeBerth** = null

Time	**Clerk** C_2
t_4	read **FreeBerth**

t_5	if **FreeBerth** is not null,
	DataForBerth = Data of P_2
	else return;

t_6	save **DataForBerth** **FreeBerth** = null

Which of the following scenarios can lead to the remaining ticket being allocated to both passengers?

○ $t_1 < t_2 < t_3 < t_4 < t_5 < t_6$

○ $t_1 < t_2 = t_4 < t_5 < t_3 < t_6$

○ $t_1 = t_4 < t_2 = t_5 < t_3 = t_6$

○ $t_4 < t_5 < t_6 < t_1 < t_2 < t_3$

We wish to find the maximum of the student marks of a class of k students as follows: we split the class into two equal batches A and B (you may assume that k is even). That is, A and B are two lists with equal number of student objects. We pass $max_A = 0$ to the first student s in A and ask s to update max_A if his/her marks is more than max_A, else do not update max_A. Then, s passes max_A to the next student in A who then does the same. This process is continued until all students in batch A are completed. At the same time, we pass $max_B = 0$ to the first student s' in B and ask s' to update max_B if his/her marks is more than max_B, else do not update max_B. Then, s' passes max_B to the next student who then does the same. This process is continued until all students in batch B are completed.

3. In the pseudocode given below, fill the blanks with the correct code from the given options such that the common code will return the maximum of the student marks.

```
/* Code at A */

  foreach (student $s_A$ in $A$) {
    **** fill code here ****
  }
```

```
/* Code at B */

  foreach (student $s_B$ in $B$) {
    **** fill code here ****
  }
```

```
/* Common Code */

  if ($max_A > max_B$) {
    return ($max_A$)
  }
  else {
    return ($max_B$)
  }
```

○
```
/* Code at A */
  if $s_A.Marks > max_A$
    $max_A = s_A.Marks$

/* Code at B */
  if $s_B.Marks > max_B$
    $max_B = s_B.Marks$
```

○
```
/* Code at A */
  if $s_A.Marks < max_A$
    $max_A = s_A.Marks$

/* Code at B */
  if $s_B.Marks < max_B$
    $max_B = s_B.Marks$
```

○
```
/* Code at A */                    /* Code at B */
  if $s_A.Marks \geq max_A$          if $s_B.Marks \geq max_B$
    $max_A = s_A.Marks$                $max_B = s_B.Marks$
```

○
```
/* Code at A */                    /* Code at B */
  if $s_A.Marks \leq max_A$          if $s_B.Marks \leq max_B$
    $max_A = s_A.Marks$                $max_B = s_B.Marks$
```

4. Suppose we divide the class into n batches $B_1, B_2, \cdots, B_n$ such that $n > 2$. Each batch B_i will return its maximum marks, which we denote by max_i. These values are passed on to the common code as a list: $[max_1, max_2, \cdots, max_n]$. The common code finds the maximum by scanning the list as follows: start with $M = 0$. If $M < max_1$, update $M = max_1$. Continue the scan to max_2 and update the maximum after comparison and so on. At the end of n comparisons, M will have the maximum value for the whole class. Let $t_1, t_2, \cdots, t_n$ be the number of comparisons done by $B_1, B_2, \cdots, B_n$ respectively, to return their maximum values. We assume that $t = t_1 = t_2 = \cdots = t_n$. From the given information, what can we say about speeding up the computation by dividing the class into more than two batches? Recall that n, the number of batches, is also the number of comparisons done by the common code.

   ○ Dividing into more than two batches will not result in any speed up.

   ○ The computation time will keep reducing as we increase the number of batches.

   ○ The computation time will keep increasing as we increase the number of batches.

   ○ Upto a limit, the computation time will reduce as we increase the number of batches. Beyond that limit, the computation time will again increase.

   ○ The computation time will vary in an unpredictable manner as we increase the number of batches.

5. Let the strength of the class be 64. If every batch can be recursively divided into two smaller batches (that is, each smaller batch can be further divided into two smaller batches), then find the total number of batches into which the class has to be divided such that we get the result in the shortest duration. You may assume the following:

   - each student takes 1 unit of time to compare their marks with the current maximum in their batch
   - the common code takes time n if the number of batches is n

   ○ 64

- ◯ 16
- ◯ 8
- ◯ 2

Given the undirected graph **G** in Figure 1, a message is sent from the vertex **s** to vertex **t** as follows: **s** will pass on the message to its neighbours in Step 1. In the next step, the neighbours of **s** will send the received message to their own neighbours and so on.

6. In how many steps will **t** receive the message?

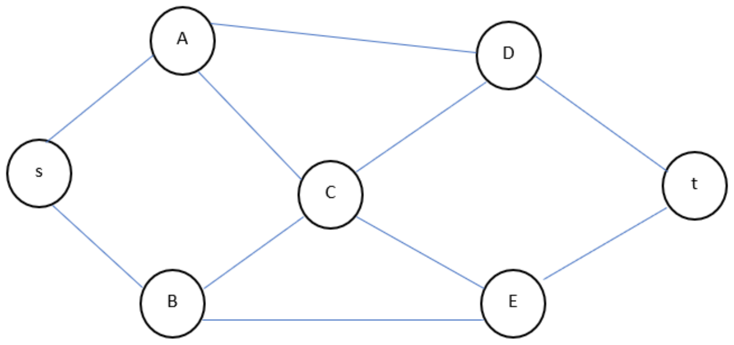

Figure 1: Graph G

- ○ 1
- ○ 2
- ○ 3
- ○ 4

7. Let us name the above technique as **Level Neighbours First (LNF)**. Now, consider sending a message from vertex **s** to vertex **t** using DFS of **G**. That is, whenever a vertex is marked as visited, it gets the message. Which do you think is faster: LNF or using DFS, and why?

- ○ LNF, because it takes the path with the minimum number of vertices to send the message.
- ○ DFS, because once the order of visiting nodes is fixed, there is only one unique path from **s** to **t**.
- ○ LNF, because a vertex will receive the message from various other vertices in parallel.
- ○ DFS, because the order of visiting nodes using DFS of a given graph is fixed.