PSEUDO CODES OF ASSIGNMENT 2

QUESTION 1

```
// Prompt the user to enter the number of batsmen
Print "Enter the number of batsmen"
Read numbatsman

// Prompt the user to enter the number of innings
Print "Enter the number of innings"
Read numinnings

// Declare a 2D array to store batting performance
Declare 2D array battingperformance[numbatsman][numinnings]

// Loop through each batsman
For i = 0 to numbatsman - 1
  Print "Enter batting performance for batsman i + 1"
  // Loop through each inning for the current batsman
  For j = 0 to numinnings - 1
    Print "Inning j + 1:"
    Read battingperformance[i][j]

// Loop through each batsman
For i = 0 to numbatsman - 1
  // Initialize variables for batting statistics
  totalruns = 0
  max = battingperformance[i][0]
  century = 0
  halfcentury = 0
  avg = 0

  // Loop through each inning for the current batsman
```

```
For j = 0 to numinnings - 1
    // Update total runs
    totalruns += battingperformance[i][j]

    // Update maximum score
    If battingperformance[i][j] > max
        max = battingperformance[i][j]

    // Check for centuries
    If battingperformance[i][j] >= 100
        century++

    // Check for half-centuries
    If battingperformance[i][j] >= 50 OR battingperformance[i][j] >= 99
        halfcentury++

// Calculate average runs
avg = totalruns / numinnings

// Print batting statistics for the current batsman
Print "Total runs by batsman i + 1: totalruns"
Print "Highest score made by batsman i + 1: max"
Print "Total centuries made by batsman i + 1: century"
Print "Total half centuries made by batsman i + 1: halfcentury"
```

QUESTION 2

```
Function min(a, b, c):
    m = a
    If m > b Then
        m = b
    If m > c Then
        m = c
    Return m

Function printMaxSubSquare(R, C, M[][]):
    Declare S[R][C]
    max_of_s = 0
    max_i = 0
    max_j = 0

    // Set the first column of S[][]
    For i = 0 to R - 1:
        S[i][0] = M[i][0]

    // Set the first row of S[][]
    For j = 0 to C - 1:
        S[0][j] = M[0][j]

    // Fill the rest of S[][] and find maximum entry
    For i = 1 to R - 1:
        For j = 1 to C - 1:
            If M[i][j] == 1 Then
                S[i][j] = min(S[i][j - 1], S[i - 1][j], S[i - 1][j - 1]) + 1
            Else
                S[i][j] = 0
            End If
            If max_of_s < S[i][j] Then
                max_of_s = S[i][j]
                max_i = i
                max_j = j
            End If

    // Print the maximum size sub-matrix
    Print "Maximum size sub-matrix is:"
    For i = max_i down to (max_i - max_of_s) + 1:
        For j = max_j down to (max_j - max_of_s) + 1:
            Print M[i][j]
        End For
        Print newline

    // Print the dimension of the largest square submatrix found
    Print "Dimension of the largest square submatrix found is: [" + max_of_s + "][" + max_of_s + "]"

Function main():
    Declare R, C, i, j
    Print "Enter the number of rows: "
    Read R
    Print "Enter the number of columns: "
    Read C
    Declare M[R][C]

    // Input elements of matrix M[][]
    For i = 0 to R - 1:
        For j = 0 to C - 1:
            Print "Element-[" + i + "][" + j + "]: "
            Read M[i][j]

    // Call the function to print the maximum sub-square matrix
    Call printMaxSubSquare(R, C, M)
    Read getchar()
```

QUESTION 3

```
// Declare constants for the array dimensions
Define R as 5
Define C as 4
```

```
// Declare and initialize the flight schedule array
Declare 2D array arr[R][C]
Initialize arr with flight schedule values

// Print information about available flights
Print "Flight is available for following days in 'MORNING':"
Print "Monday, Tuesday, Thursday, Friday"
Print "Best option for booking a flight is on: Monday (due to price factor)"

Print "\nFlight is available for following days IN 'EVENING':"
Print "Tuesday, Wednesday, Friday"
Print "Best option for booking a flight is on: Wednesday (due to price factor)"

// Prompt the user to enter the day for detailed information
Print "\nEnter M for information about flight on 'Monday', T for 'Tuesday', W for 'Wednesday', H for 'Thursday', F for 'Friday':"
Read day

// Use a switch statement to provide detailed information based on the user's input
Switch (day):
  Case 'M':
    Print "Flight is available in Morning, price(300)"
    Break
  Case 'T':
    Print "Flight is available in Morning, price(320) as well as in Evening, price(310)"
    Break
  Case 'W':
    Print "Flight is available in Evening, price(280)"
    Break
  Case 'H':
        Print "Flight is available in Morning, price(380)"
        Break
    Case 'F':
        Print "Flight is available in Morning, price(375) as well as in Evening, price(400)"
        Break
    Default:
        Print "Invalid day"

// Prompt the user to enter the preferred time and price
Print "\nEnter time which suits you ('M' for Morning and 'E' for evening):"
Read time
Print "Enter most suitable price for you:"
Read price

// Use if-else statements to provide booking suggestions based on user's preferences
If (time is 'M'):
    If (price is 300):
        Print "You can book a flight on Monday (Morning)"
    Else if (price is 320):
        Print "You can book a flight on Tuesday (Morning)"
    Else if (price is 380):
        Print "You can book a flight on Thursday (Morning)"
    Else if (price is 375):
        Print "You can book a flight on Friday (Morning)"
    Else:
        Print "Invalid value"
Else if (time is 'E'):
    If (price is 310):
        Print "You can book a flight on Tuesday (Evening)"
    Else if (price is 280):
        Print "You can book a flight on Wednesday (Evening)"
    Else if (price is 400):
        Print "You can book a flight on Friday (Evening)"
    Else:
        Print "Invalid value"
Else:
    Print "Invalid value"
```

QUESTION 4

```
// Prompt the user to enter the number of rows and columns
Print "Enter the number of rows: "
Read r
Print "Enter the number of columns: "
Read c

// Declare a 2D array to represent the maze
Declare 2D array maze[r][c]

// Prompt the user to enter elements for the maze
Print "Enter elements ('W' for walls, 'O' for open paths, 'S' for start, 'E' for exit):"
For i = 0 to r - 1:
    For j = 0 to c - 1:
        Print "Character-[" + i + "][" + j + "]: "
        Read maze[i][j]

// Print the original maze
Print "\nOriginal maze:"
For i = 0 to r - 1:
    For j = 0 to c - 1:
        Print maze[i][j] + " "
    Print newline

// Find the starting position (S) in the maze
```

```
For i = 0 to r - 1:
    For j = 0 to c - 1:
        If maze[i][j] is 'S':
            start = i
            end = j
            Break

// Initialize variables for current position in the maze
currstart = start
currend = end

// Print the transversal of the maze
Print "\nTransversal maze:"
While maze[currstart][currend] is not 'E':
    Print "(" + currstart + "," + currend + ") "
    If currstart < r - 1 AND maze[currstart + 1][currend] is not 'W':
        Increment currstart
    Else if currend < c - 1 AND maze[currstart][currend + 1] is not 'W':
        Increment currend
    Else:
        Print "Transversal maze can't be found due to constraint"
        Break

Print "(" + currstart + "," + currend + ") "
Print newline

Return 0
```

QUESTION 5

```
Function printRamanujan(n):
    For i from 1 to n:
        For j from 1 to n:
```

```
      For k from i + 1 to n:
        For l from k to n:
          If i^3 + j^3 equals k^3 + l^3 Then
            Print i^3 + j^3 = k^3 + l^3 = (i^3 + j^3)
          End If
        End For
      End For
    End For
  End For

Function main():
  Declare n
  Print "Enter the value of n: "
  Read n

  Print "Ramanujan-Hardy numbers less than n^3:"
  Call printRamanujan(n)

  Return 0
```

QUESTION 6

```
Function main():
  Declare N, T
  Print "Enter the size of your array:"
  Read N

  Print "Enter the value of your targeted sum:"
  Read T

  Declare array arr[N]

  // Input elements of the array
  For i from 0 to N - 1:
    Print "Enter the value of element " + (i + 1) + ":"
    Read arr[i]

  Declare pair as 0

  // Find pairs that sum up to the targeted value
  For i from 0 to N - 1:
    For j from 0 to N - 1:
      If arr[i] + arr[j] equals T Then
        Print "These pairs make the sum T: (" + arr[i] + ", " + arr[j] + ")"
        Set pair to 1
      End If
    End For
  End For

  // Check if no pair was found
  If pair is not 1 Then
    Print "No pairs found that make the targeted sum " + T + ". Try different inputs."
  End If

  Return 0
```

QUESTION 7

```
// Function to swap two integers
Function swap(a, b):
  temp = a
  a = b
  b = temp

// Function to perform bubble sort on ages and prices
Function bubbleSort(ages[], prices[], n):
    For i from 0 to n - 1:
        For j from 0 to n - i - 1:
            // Sort based on age in ascending order
            If ages[j] > ages[j + 1] Then
                Call swap(ages[j], ages[j + 1])
                Call swap(prices[j], prices[j + 1])
            Else If ages[j] == ages[j + 1] Then
                // If ages are equal, sort based on price in descending order
                If prices[j] < prices[j + 1] Then
                    Call swap(prices[j], prices[j + 1])
                End If
            End If
        End For
    End For

// Function to print the sorted lists
Function printSortedLists(ages[], prices[], n):
    Print "Sorted List in Ascending Order based on Age:"
    Print "Age\tPrice"
    For i from 0 to n - 1:
        Print ages[i] + "\t" + prices[i]

    Print "\nSorted List in Descending Order based on Price (within the same age):"
    Print "Age\tPrice"
    For i from 0 to n - 1:
        Print ages[i] + "\t" + prices[i]

// Main function
Function main():
    Declare ages as array
    Declare prices as array
```

```
   Declare n

   // Initialize ages and prices arrays
   ages = {24, 25, 20, 28, 27}
   prices = {21, 30, 19, 28, 35}
   n = length of ages or prices

   Print "Original List:"
   Print "Age\tPrice"
   For i from 0 to n - 1:
      Print ages[i] + "\t" + prices[i]

   // Call bubbleSort function to sort the lists
   Call bubbleSort(ages, prices, n)

   // Call printSortedLists function to print the sorted lists
   Call printSortedLists(ages, prices, n)

   Return 0
```

QUESTION 8

```
Function main():
   Declare n
   Print "Enter a positive integer (or EOF to exit):"

   // Check if the given value from the user is not EOF
   While (Read n) is not EOF:
      // Check if the entered value is negative
      If n < 0 Then
         Print "Enter a positive integer, not negative"
         Continue to the next iteration of the loop
```

```
        Declare persistence as 0

        // Calculate the persistence of the entered number
        While n >= 10:
            Declare product as 1

            // Calculate the product of digits
            While n > 0:
                product *= n % 10
                n /= 10 // Remove the last digit of the number

            Set n to product
            Increment persistence

        Print "Persistence of the entered number is " + persistence
        Print "Enter a positive integer (or EOF to exit):"

    Return 0
```

QUESTION 9

```
Function spiralOfMatrix(row, col, arr1):
    Declare i, rows, column
    Set rows to 0
    Set column to 0

    While rows is less than row AND column is less than col:
        // Print the top row
        For i from column to col - 1:
            Print arr1[rows][i]

        Increment rows
        // Print the rightmost column
        For i from rows to row - 1:
            Print arr1[i][col - 1]

        Decrement col

        // Print the bottom row (if exists)
        If rows is less than row:
            For i from col - 1 to column:
                Print arr1[row - 1][i]

            Decrement row

        // Print the leftmost column (if exists)
        If column is less than col:
            For i from row - 1 to rows:
                Print arr1[i][column]

            Increment column

// Main function
Function main():
    Declare i, j
    Declare arr1 as a 2D array of size R x C

    // Initialize the array
    arr1 = {{1, 2, 3, 4, 5},
            {6, 7, 8, 9, 10},
            {11, 12, 13, 14, 15},
            {16, 17, 18, 19, 20}}
```

```
Print "The given array in matrix form is:"
For i from 0 to R - 1:
    For j from 0 to C - 1:
        Print arr1[i][j] + " "
    Print newline

Print "The spiral form of the above matrix is:"
Call spiralOfMatrix(R, C, arr1)

Return 0
```