Day 4

```
1) procedure
 declare two variables a and b
 set a to non-zero value
 set b to 0
 try block:
 attempt to divide a by b
result in another variable result
exception:
 catch the Arithemetic Exception
 display an error message
 end try-catch block
end procedure
2) Array Bound of exception
 procedure
 initialize arr[] := {5,6,7,8,9}
set a variable index to a value greate
 than the array's length
 try block:
 Attempt to access the array element
 index
 exception:
 catch Array Index out of Bound Exception
   display Array message
```

) use a hashmap
pseudo code

```
) . procedure
  group-anagrams (string s: list
 of string) -> list of list of strings
    initialize a HashMap
      anagram_map
       sorted _s = sort(s)
       anagram_map [sorted _s] . append(s)
      anagram_map [sorted_s] = [s]
       list (anagram _map. values())
```

4) pseudo code

```
  procedure
   n = length of numbers
   expected_sum = n * (n+1) / 2
   actual_sum = sum (numbers)
   missing_number = expected_sum
                      - actual_sum
     return missing_number.
```

5) pseudo code procedure

```
abstract class statistics:
abstract function total(data: list of numbers)
abstract function average(data: list of numbers:)
function mean (data: list of numbers):
    total_sum = sum(data)
    count = length of data
      return total_sum/count
```

6) pseudo code for creating an interface with 4 methods (add, sub, mul, div)

procedure

```
interface calculator:
 function add (a: number, b: number)
function sub(a: number, b: number)
function mul ((a: number, b: number)
 function div(a: number, b: number)
class simplecalculator implements
            calculator:
```

```
function add (a: number, b: number)
    return a+b
function sub (a: number, b: number)
    return a-b
function mul (a: number, b: number)
    return a*b
function div (a: number, b: number)
    return a/b
```

7) pseudo code : for creating 3 interface

```
procedure:-
interface sumcalculator;
    function sum (data: list of numbers)
interface Avgcalculator:
    function avg (data: list of numbers)
interface percentage calculator;
    function percentage (obtained
        _marks: number, total_marks: number)
    total_sum = 0
    for each number in data:
    total_sum = total_sum + number
    return total_sum
```

```
result-sum = calc.sum(data)
print.("sum:", result-sum)
result._avg := calc.avg.(data)
print (:"avg:"_ result-avg)
result- percentage := calc.perce
                     ntage.(data)
print.("percentage:", result-
                     percentage:)
```

8) pseudo code for. creating an interface

```
 procedure:
interface Tree;
 function.fruits (quantity: number,
                type: string)
 function. leaves. (quantity.: number
     , type:: string:, colour: string)
function. flowers (quantity, number
          , type: string, color:
                    string, season:
    string:)
 implements Tree:
fuction fruits. (quantity: number
     , type: string) :
```

·int ("Branch 1 has", quantity, type,
      "fruits.")

function flowers (quantity: number,
      type: strings. colour: string,
      season: string):
   print ("Branch 1 has", quantity,
      type, "flowers of", colour, "color
      that bloom in", season, ".")

9) pseudo code to demonstrate
   how to find the union intersec

procedure:

```
function set operation():
set 1 = create set ([1, 2, 3, 4, 5])
set 2 = create set ([4, 5, 6, 7, 8])
union set = create set (set1)
 union-set. addAll. (set 2)
print ("union of set 1 and
          set2 :", union _ set)
```

```
intersection_set = (reat.set (.set1)
intersection_set.retainAll (set2)
print ("intersection of set1 and
                        set2")
difference_set = create

10). procedure
  declare.scanner class
    initialize base = 4
           exponent = 3
    using the intuit function
    assign.pow =. Mather.power
                 (base, exponent)
     display. the result
    end procedure
```