HL Unit 5 – Abstract Data Structures

Quiz 2 – 2D Arrays

Question 1			
Objectives:	5.1.4, 5.1.5	Exam Reference:	Nov-14 12

A population study divides a metropolitan area into seven regions: A–G.
The following table shows the current population (in millions) of the regions.

Region	Current population (millions)
A	2.3
B	2.1
C	1.2
D	1.4
E	1.5
F	1.1
G	0.8

Two one-dimensional arrays, *Region* and *Curr_Pop*, are used to hold this data.
For example, *Region[0] ='A'*. The population in region *A* is 2.3 million and 2.3 is found in *Curr_Pop[0]*.

(a) Construct the algorithm that will output the total population in the metropolitan area. [3]

Award marks as follows.
*Award **[1]** for a correct loop (accept a correct while loop).*
*Award **[1]** for correct calculation.*
*Award **[1]** for correct output.*

```
totalpop = 0
loop K from 0 to 6
     totalpop = totalpop + Curr_Pop[K]
end loop
output totalpop
```

*Note: Award **[1]** for `Curr_Pop[0] + Curr_Pop[1]+ … + Curr_Pop[6]`*
Award marks only for an algorithm, do not award marks for calculation / summation "2.3 + 2.1 + 1.2..."

The numbers in the following table represent expected percentages of yearly migration from one region to another, obtained by analysing historical migration data. For example, it is expected that 0.32 % of the current population of region B will move to region C.
The diagonal entries represent a region's internal growth rate. For example, the population of region C is expected to increase by 1.2 % as a result of the births and deaths of people currently living in region C.

To / **From**	A	B	C	D	E	F	G
A	**1.10**	0.21	0.21	0.05	0.20	0.20	0.29
B	0.30	**1.20**	0.32	0.25	0.20	0.09	0.31
C	0.25	0.22	**1.20**	0.35	0.30	0.23	0.12
D	0.10	0.33	0.36	**1.30**	0.09	0.12	0.20
E	0.20	0.22	0.24	0.35	**1.00**	0.20	0.21
F	0.12	0.21	0.13	0.21	0.22	**1.40**	0.31
G	0.05	0.03	0.30	0.20	0.23	0.26	**0.90**

(b) (i) State the percentage of the population of region *G* that are expected to move to region *A*. [1]

0.05 (%);

(ii) Determine the number of people from region *B* who are expected to move to region *E*. [1]

0.0042 (million) / 4200;
*Award **[1]** for 0.2 * 2.1 /100*.

(iii) Describe how the change in population of region F in one year could be determined. [3]

*Award up to **[3 max]**.*
Current population in the region F should be increased by the internal growth rate;
Increased by the population who moved into the region;
And decreased by the population who moved from this to other regions;
Accept formulas that correctly address these points.

(c) Construct the algorithm that will predict the population in each region after 10 years. You should assume that the yearly migration percentages, given in the table on page 8, remain the same over the 10 years. [7]

*Award marks as follows up to **[7 max]**.*
*Award **[1]** for correct outer loop (10 years).*
*Award **[1]** for correctly initializing Future_Pop array to zero each year.*
*Award **[1]** for correct row loop (k).*
*Award **[1]** for correct column loop (j).*
*Award **[1]** for correctly calculated internal growth.*
*Award **[1]** for correct calculation of population who moved to region, increment.*
*Award **[1]** for correct calculation of population who moved from region, decrement.*
*Award **[1]** for assignment (new data in Curr_Pop array is data from Future_Pop array).*

Example answer:

```
loop YEAR from 1 to 10
  loop Z from 0 to 6
    Future_Pop[Z]=0
  end loop
  loop K from 0 to 6
    loop J from K to 6
      if J=K then
        Future_Pop[K] = Curr_Pop[K] * Table[K][K]
      else
        Future_Pop[K]=Future_Pop[K]
                        +(Curr_Pop[J]*Table[J][K])
                          -(Table[K][J]*Curr_Pop[K])
      end if
    loop Z from 0 to 6
      Curr_Pop[Z] = Future_Pop[Z]
    end loop
  end loop
end loop
loop Z from 0 to 6
  output Curr_Pop[Z]
end loop
//Output the contents of Curr_Pop array which holds the population
//after 10 years
```

Question 2			
Objectives:	5.1.4, 5.1.5	Exam Reference:	Nov-16 12

A two-dimensional array, *A*, has *N* rows and *N* columns, where *N* is a positive integer.

The following algorithm is written to fill array *A* with the numbers 1, 2, 3,…, *N2*.

```
N=input('Enter an integer greater than zero')
K=N*N
loop for ROW=0 to N-1
  loop for COLUMN=0 to N-1
       A[ROW][COLUMN]=K
       K=K-1
    end loop
end loop
```

(a) Trace the algorithm, with an input of *N=3*, to show the contents of array *A* after the algorithm has been executed. [3]

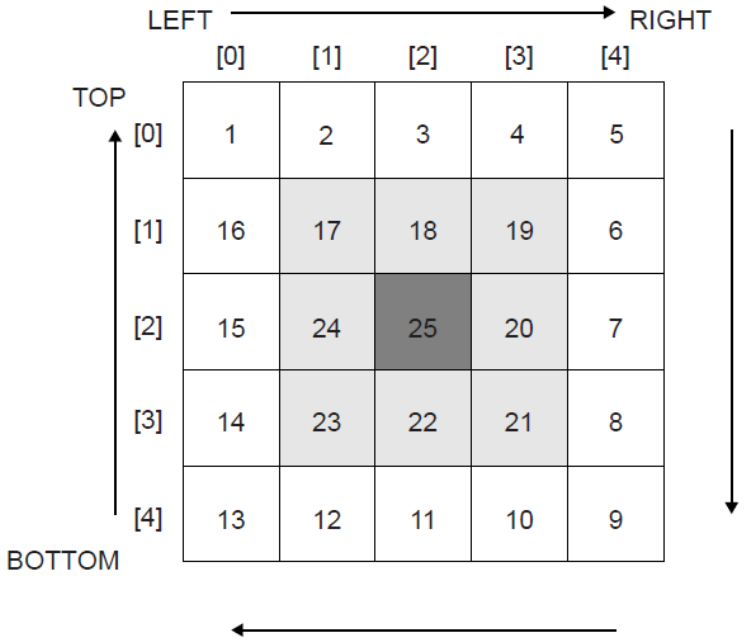
*Award **[1]** for each correct row, up to **[3 max]**.*

	[0]	[1]	[2]
[0]	9	8	7
[1]	6	5	4
[2]	3	2	1

Accept answers that transpose the table.

There are many different ways of placing the numbers 1 to N^2 into an $N \times N$ two-dimensional array.

The following two-dimensional array, with dimensions 5 × 5 has been filled in a circular (spiral) pattern with numbers 1 to 5^2.



The general process of filling an $N \times N$ two-dimensional array, in a circular (spiral) pattern, with numbers from 1 to N^2 could be described as follows:

- initialize *Z=1*,
- initialize *TOP*, *BOTTOM*, *LEFT* and *RIGHT*,
- iterate until the whole array is filled,
- each time *Z* is placed correctly increase the value of *Z* by 1,
- fill the elements of the *TOP* row starting from *LEFT* to *RIGHT*,
- increase *TOP* by 1 before filling the elements of the *RIGHT* column,
- fill the elements of the *RIGHT* column starting from *TOP* to *BOTTOM*,
- decrease *RIGHT* by 1 before filling the elements of the *BOTTOM* row,
- and continue filling the *BOTTOM* row and *LEFT* column in a similar way, adjusting *TOP*, *RIGHT*, *BOTTOM* and *LEFT* accordingly.

(b) (i) State the initial values for *TOP*, *BOTTOM*, *LEFT* and *RIGHT*. [1]

*Award **[1]** only for all correct values.*

```
TOP=0
BOTTOM=N-1
LEFT=0
RIGHT=N-1
```

(ii) State the consequence of not increasing *TOP* by 1 before starting to fill the *elements of the RIGHT column*. [1]

The array element at position `[TOP][RIGHT]` in which value of `Z` is already placed, will be overwritten by the value of `Z + 1`;
Not all of the numbers 1 to N^2 will be placed in the array because some will be overwritten;
The array will be filled with more than N^2 numbers/with numbers greater than N^2;

Accept answers from the sample 5×5 table, eg the value of `MATRIX[0][4]` which is already filled by 5, will be changed to 6.

(iii) In the algorithm described above, state the indices (subscripts) of the first and the last element to be filled in the *BOTTOM* row. [1]

The first element to be filled in `BOTTOM` row has indices (subscripts) `[BOTTOM][RIGHT]` and the last to be filled has indices (subscripts) `[BOTTOM][LEFT]`;

Accept answers from the sample 5 × 5 table. The first element to be filled in `BOTTOM` row has indices (subscripts) [4][3] and the last to be filled has indices (subscripts) [4][0].

(c) Construct, in pseudocode, an algorithm to fill an $N \times N$ two-dimensional array, in a circular (spiral) pattern, with numbers from 1 to N^2 as described above. [9]

*Award up to **[9 max]** as follows.*
*Award **[1]** for initializing `Z`.*
*Award **[1]** for initialization of the top and bottom rows, and left and right columns.*
*Award **[1]** for the outer loop (must be `while`).*
*Award **[1]** for the idea that four inner loops are needed (could be `for` or `while` loops).*
*Award **[1]** for each correct inner loop up to **[4 max]**.*
*Award **[1]** for assignment (current value of `Z` placed in `A`).*
*Award **[1]** for changing the value of `Z` after each assignment.*
*Award **[1]** for changing values of `TOP`, `BOTTOM`, `LEFT`, `RIGHT`.*

Example answer 1:

```
Z=1
TOP=0
BOTTOM=N-1
RIGHT=N-1
LEFT=0
loop while Z<=N*N
  COUNT1 = LEFT
  loop while COUNT1 <= RIGHT
    A[TOP][ COUNT1] = Z
    Z = Z+1
    COUNT1 = COUNT1+1
  end loop
  TOP = TOP+1
  COUNT2 = TOP
  loop while COUNT2 <= BOTTOM
    A[COUNT2][ RIGHT] = Z
    Z = Z+1
    COUNT2 = COUNT2+1
  end loop
  RIGHT = RIGHT-1
  COUNT3 = RIGHT
  loop while COUNT3 >= LEFT
    A[BOTTOM][ COUNT3] = Z
    Z = Z+1
    COUNT3 = COUNT3-1
  end loop
  BOTTOM = BOTTOM-1
  COUNT4 = BOTTOM
  loop while COUNT4 >= TOP
    A[COUNT4][ LEFT] = Z
    Z = Z+1
    COUNT4 = COUNT4-1
  end loop
  LEFT = LEFT+1
end loop WHILE
```

Example answer 2:

```
Z=1
TOP=0
BOTTOM=N-1
RIGHT=N-1
LEFT=0
loop while Z<=N*N
  loop for i from LEFT to RIGHT
    A[TOP][i]=Z
    Z=Z+1
  end loop
  TOP=TOP+1
  loop for i from TOP to BOTTOM
    A[i][RIGHT]=Z
    Z=Z+1
  end loop
  RIGHT=RIGHT-1
  loop for i from RIGHT downto LEFT
    A[BOTTOM][i]=Z
    Z=Z+1
  end loop
  BOTTOM=BOTTOM-1
  loop for i from BOTTOM downto TOP
    A[i][LEFT]=Z
    Z=Z+1
  end loop
  LEFT=LEFT+1
end loop WHILE
```

Note: *For both examples, assume that integer* `N` *is inputted and the space for the array* `A` *with dimensions* $N \times N$ *is allocated.*