CS1010E Programming Methodology
Tutorial 8: 2D Arrays

> Being ignorant is not so much a shame, as being unwilling to learn.
>
> ~ *Benjamin Franklin*

To students:

Please be reminded that the submission deadline for **Problem Set 3** is **March 23rd.**

I. Manual Tracing

1. For each of the program fragments below, write down its output.

   (a) [CS1010E AY2010/2011 Semester 2 Exam, Q4]

```
int arr[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
int i = 0, j = 0;

for(i = 0; i < 3; i++) {
  for(j = 0; j < 3; j++) {
    printf("%d ", arr[j][i]);
  }
}
printf("\n");
```

   (b) [CS1101C AY2008/2009 Sem1 Exam, Q10]

```
int a[4][4] = {{0, 1, 2, 3}, {1, 2, 3, 0},
               {2, 3, 0, 1}, {3, 0, 1, 2}};
int b[4], i;

for (i = 0; i < 4; i++) {
  b[i] = a[a[i][i]][a[i][i]];
}

printf("%d %d %d %d\n", b[0], b[1], b[2], b[3]);
```

(c) [CS1101 AY2008/2009 Semester 1 Exam, Q4]

```
int array[3][3] = { {1, 1, 1}, {2, 2, 2}, {3, 3, 3} };
int i, j;

for (i = 0; i < 3; i++) {
  for (j = 0; j < 3; j++) {
    array[i][j] += array[j][i];
  }
}

for (int i = 0; i < 3; i++) {
  for (int j = 0; j < 3; j++) {
    printf("%d ", array[i][j]);
  }
  printf("\n");
}
```

II. Hands-on Session

2. Study the following program.

```
#include <stdio.h>
#define MAX_ROW 2
#define MAX_COL 3

void print_array(int arr[MAX_ROW][MAX_COL], int row_size, int
col_size);

int main(void) {

  int values[MAX_ROW][MAX_COL];
  int row, col;

  printf("Enter values:\n");
  for (row = 0; row < MAX_ROW; row++) {
    for (col = 0; col < MAX_COL; col++) {
      scanf("%d", &values[row][col]);
    }
  }

  printf("Array entered contains:\n");
  print_array(values, MAX_ROW, MAX_COL);

  return 0;
}
```

```
// Print elements in array arr
void print_array(int arr[MAX_ROW][MAX_COL], int row_size, int
col_size) {

  int row, col;

  for (row = 0; row < row_size; row++) {
    for (col = 0; col < col_size; col++) {
      printf("%d ", arr[row][col]);
    }
    printf("\n");
  }
}
```

Enter the data in each of the following formats. Do they work? What can you deduce?

```
8 1 2
3 0 9
```

```
  8  1   2
 3  0     9
```

```
8 1
2 3
0 9
```

```
8
1 2 3

0 9
```

```
8
1
2
3
0
9
```

3. **[Problem Set 3 Exercise #24] Square Matrix**

4. **Input and output redirection**

   If the array is big it might be too troublesome to key in all its elements interactively. One approach is to use a file to store the input data and run the program to read values from this input file. This will be covered in Q6 later.

   Another simple trick is to use the input redirection feature '**<**' in UNIX. Create an input text file called **matrix.in** with the following content using **vim**:

```
5
2 -1 3 4 1
0 7 5 -2 0
0 0 6 0 4
0 0 0 0 8
0 0 0 0 2
```

   When you run your matrix program in Q3 above, you just need to type this:

```
a.out < matrix.in
```

   The program will read data from matrix.in as if you have entered these data on the keyboard. Try it out.

   You may also do output redirection '**>**' in UNIX. Try out the following commands.

```
a.out < matrix.in > outfile
vim outfile
```

5. **Simple file processing**

   Type the following command to make a copy of your program written in Q3.

```
cp square_matrx.c square_matrx_v2.c
```

   Your program **square_matrix_v2.c** will read data from an input file **matrix.in**, using file processing functions provided by C language:

   A sample run is given below:

```
Enter the size of the square matrix: Enter elements
row by row:
Given matrix is not a diagonal matrix.
Given matrix is an upper triangular matrix.
```