EE101 C programming and SW engineering 1
Lab Practice 9 – Pointers, Arrays & Dynamic Memory Allocation

Use your preferred compiler to investigate the programming exercises below. This laboratory concerns programs and functions that use multidimensional arrays. It will develop your understanding of memory allocation and de-allocation.

Exercise 1

Analyse and explain in detail the following program.

```
#include<stdio.h>
#include<stdlib.h>              /*for malloc( ), free( ), exit( ) and EXIT_FAILURE*/

int main( )
{
double * ptd;
int max, number, i = 0;

puts("What is the number of type double entries?");
scanf("%d", &max);
ptd = (double *) malloc(max * sizeof (double));

if(ptd == NULL){
        puts("Memory allocation failed. Goodbye.");
        exit(EXIT_FAILURE);        /*EXIT_FAILURE is a special value in C*/
        }
/* ptd now points to an array of max elements */

puts("Enter the values (q to quit):");

while(i < max && scanf("%lf", &ptd[i]) == 1)
        ++i;

printf("Here are your %d entries:\n", number = i);

for(i = 0; i < number; i++){
        printf("%7.2f", ptd[i]);
        if(i % 7 == 6)
                putchar('\n');
        }

if(i % 7 != 0)
        putchar('\n');

puts("Done.");
free(ptd);
return 0;
}
```

Exercise 2

Write a program that declares a 3 by 5 two-dimensional array and initialises it with some values of your choice. The program should:

a. print all the values;
b. double the values;
c. print all the values again.

Write a function that is responsible for displaying the values and a second function that doubles the values. The functions could take as arguments the **array name** and the **number of rows** as arguments.

Exercise 3

Write a program that determines prime numbers using the Eratosthenes' method, which works as follows:

To find all the prime numbers less than or equal to a given integer n:

1. Create a list of consecutive integers from 2 to n: (2, 3, 4, ..., n).
2. Initially, let p equal 2, the first prime number.
3. Strike from the list all multiples of p greater than p.
4. Find the first number remaining on the list greater than p (this number is the next prime); let p equal this number.
5. Repeat steps 3 and 4 until p is greater than n.
6. All the numbers remaining on the list are prime.

The number of integers to be considered will be given as an input. The program will dynamically allocate the required memory for the integer array. If there is not enough memory available, the program will ask the user to ask for a smaller value. Use the malloc( ) function for that purpose. Here is a sample run of the program:

```
How many integers do you want to examine: 200
Between 1 and 200, the prime numbers are:

2      3      5      7      11     13     17     19     23     29
31     37     41     43     47     53     59     61     67     71
73     79     83     89     97     101    103    107    109    113
127    131    137    139    149    151    157    163    167    173
179    181    191    193    197    199
```