MIDDLE EAST TECHNICAL UNIVERSITY, NORTHERN CYPRUS CAMPUS
CNG140 C Programming – Worksheet 18

Purpose:
In this work sheet, you will use practice pointers, pointers and functions and in particular you will exercise pass by reference.

Tasks:

a. Write a program that includes the following function that determines the smallest number of $20, $10, $5, and $1 necessary to pay the amount represented by the dollars parameter.

```
void pay_amount(int dollars, int *twenties, int *tens, int *fives, int *ones)
```

The twenties parameter points to the variable in which the function will store the number of $20 bills required. The tens, fives and ones parameters are similar. Your main function will get the amount to pay from the user and it will also maintain variables to that will keep track of the minimum number of twenties, tens, fives and ones that is needed pay the represented amount. These will also be initialized to zero. Their addresses will then be sent to this function such that they are updated by this function (pay_amount) accordingly. Once this function call finishes, your main function will display the values of twenties, tens, fives and ones.

A sample run would be as follows:

```
Enter the amount to pay: 95
You need to pay 4x$20 + 1x$10 + 1x$5
```

b. Write a program that includes two matrix operations given below. Your program should include the following functions. Imagine that your porgram will have matrix A and B as follows:

$$A = \begin{matrix} 1 & 9 & 6 \\ 5 & 9 & 7 \\ 4 & 8 & 4 \end{matrix} \qquad B = \begin{matrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{matrix}$$

1. In your main program ask the number of rows and columns of your matrix, and create two matrices accordingly. For example, if the user says 3 rows and 3 columns, then A, B will be created accordingly. We will assume that both A and B are the same sizes.
2. Write a function called `populate()` which can populate a given matrix with random numbers between 0-10 → input: two dimensional array, number of rows and number of columns, output: void;
3. Write a function called `multiplication()` that computes A*B → input: array A, B and result array, number of rows, number of columns, output: void
4. Write a function called `display()` to display a matrix –> input: array, number of rows, number of columns, output: void.

In your main you will need to declare two dynamic arrays. You will then populate these arrays with random numbers by using populate function. You also need to create an array dynamically with the same size which is called result where all values are initialised to zero. You will then call the subtraction function above to calculate the results. Finally, your main will display the content of the two arrays and also the result array.

A sample run would be as follows:

```
Enter number of rows: 3
Enter number of columns: 3

Your Matrices are as follows:
============================
Matrix A is as follows:
|1 9 6|
|5 9 7|
|4 8 4|

Matrix B is as follows:
|1 4 7|
|2 5 8|
|3 6 9|

Results:
============================

A * B is as follows:
|37 85 133|
|44 107 170|
|32 80 128|
```

c. Add another function to the program you wrote in (b) as follows: Write a function called `complexAddition()` to compute A+2*sin(B) → input: array A, B and result array, number of rows, number of columns, output: void
Sampler un will be as follows:

A sample run would be as follows:

```
Enter number of rows: 3
Enter number of columns: 3

Your Matrices are as follows:
============================
Matrix A is as follows:
|1 9 6|
|5 9 7|
|4 8 4|

Matrix B is as follows:
|1 4 7|
|2 5 8|
|3 6 9|

Results:
============================

A * B is as follows:
```

```
|37 85 127|
|44 107 163|
|32 80 124|

A + B * SinB is as follows:
|1.52 10.15 7.77|
|5.63 10.41 9.19|
|4.73  9.67 6.61|
```

d. Write a program that will create a NxN array A and populate it with the user input. Value of N will be entered by the user. Then your program should convert all the array elements to a range 0 and 1.

1. Write a function called `populateArray()` which can populate a given array with the user input.
   → input: two dimensional array, number of rows and number of columns output: void;

2. Write a function called `conversion()` to convert all the array elements to a range 0 and 1 using the formula below.
   → input: two dimensional array, number of columns and number of rows output: void

$$cx = \frac{x - \min(A)}{\max(A) - \min(A)}$$

Please note that in this formula,

min(A) is the minimum element in the array A
max(A) is the maximum element in the array A
x is the element of the array A
cx is the converted element x of array A

3. Write a function called `display()` to display the array
   → input: two dimensional array, number of rows and number of columns output: void.

A sample run based on the above example would be as follows:

```
Enter the size of N: 3

Enter a value to be stored for A: 10
Enter a value to be stored for A: 20
Enter a value to be stored for A: 300
Enter a value to be stored for A: 88
Enter a value to be stored for A: 25
Enter a value to be stored for A: 101
Enter a value to be stored for A: 208
Enter a value to be stored for A: 39
Enter a value to be stored for A: 98

Created A is as follows:
    10    20   300
    88    25   101
   208    39    98
```

```
Converted A is as follows:
    0.0000    0.0345    1.0000
    0.2690    0.0517    0.3138
    0.6828    0.1000    0.3034
```

e. A barcode scanner for Universal Product Codes (UPC) verifies the 12-digit code that was scanned by comparing the code's last digit (called a check digit) to a computed value of the check digit from the first 11 digits as follows:

1. Calculate the sum of the digits in the odd-numbered positions (the first, the third, ..., eleventh digit) and multiply this sum by 3.
2. Calculate the sum of the digits in the even-numbered positions (the second, the fourth, ... tenth digit) and add this to the previous result.
3. If the last digit of the result from step 2 is 0, then 0 is the check digit. Otherwise, subtract the last digit from 10 to calculate the check digit.
4. If the check digit matches the final digit of the 12-digit UPC, then the UPC is assumed to be correct.

Write a program that includes the following functions to test if a barcode is valid or not.

1. readBarcode(int *barcode, int n) – this will read a barcode from the user and store the value in the barcode array.
2. validateBarcode(int *barcode, int n, int *checkDigit) – this will read a barcode, calculate the checkdigit as it is given above and it will return 1 if the barcode is validated and it will return 0 if it is not validated.

A sample run would be as follows:	A sample run would be as follows:
`Enter a barcode: 490125840581` `UPC is not correct!`	`Enter a barcode: 490124840581` `UPC is correct!`

f. Write a program for a simple letter game. In this game, first computer will ask user to enter the number of letters (N). Then, the computer dynamically creates a character array with size N. Computer will generate N random integers between 65 and 90 (representing the ASCII code of letters between A and Z) and then populates the character array with the corresponding letters.

For example, when N=10 and the randomly generated integers are as follows,

Randomly Generated integers:	80	72	81	71	72	85	77	69	65	89

Then the populated character array should be as follows;

Character array:	P	H	Q	G	H	U	M	E	A	Y

Then the computer will provide the user five tries to guess any duplicate characters. If the user guesses the correct letter, the message "Congrats! You win!" should be

displayed. After each incorrect guess, your program has to display the message "Wrong character, try again!" and allow the user to guess again. After five incorrect guesses, if there is any duplicate letter then computer should display the message "Sorry! You lose!" and the correct letter (If there are more than one duplicate letters, its ok to just display one). Otherwise, computer should display "There are not any duplicates! You did not win or lose!".

Define the following functions:

- `void CreateArray (char *, int)`: will populate character array with random letters. → input: one dimensional array, array size(N: number of letters) output: void.
- `int CheckLetters (char *, char , int )`: will find if there are more than one character letter given in the character array. If there are more than one of the given character, it will return 1, otherwise it will return 0. → input: one dimensional array, given letter, array size(N: number of letters) output: 0 or 1.
- `void DisplayDuplicateLetter(char *, int)`: will try to find the duplicate letter and show an appropriate message (see above definition). → input: one dimensional array, array size(N: number of letters) output: void.

A sample run based on the above example would be as follows:

```
Please enter the number of letters (N)? 10

Guess the duplicate letter: A
Wrong character, try again!
Guess the duplicate letter: M
Wrong character, try again!
Guess the duplicate letter: Y
Wrong character, try again!
Guess the duplicate letter: P
Wrong character, try again!
Guess the duplicate letter: C
Sorry! You lose!
The duplicate letter: H
```

Recommended Reading: Chapter 7 (section 7.3, p. 343-364) and Chapter 11 (section 11.3 p. 553-562)
Recommended Exercises: Programming Exercises given in the following pages.