Department of Information Systems and Technologies

CTIS 152 – Algorithms and Data Structure

SPRING 2025 - 2026

Lab Guide #1 – Week 2-1

OBJECTIVE : General Review of 151 Subjects
Instructors: Serpil TIN **Assistants :** Efe Mert Şahinkoç, Hatice Zehra YILMAZ

Q1. Write a C program for a **tennis training scoreboard**. The program will take two **positive** integers from the user (representing the **ace counts** of Player A and Player B) and then call a **swap** function using the **call-by-reference** method to exchange these values. Your program must check input validity and print an error message if a value is not positive.

Project Name: LG01_Q1

File Name: Q1.cpp

Example Run:

```
Enter Player A ace count: -9
Player A ace count must be positive.
Enter Player A ace count: 2

Enter Player B ace count: -3
Player B ace count must be positive.
Enter Player B ace count: 6

Before swap: a = 2, b = 6
After swap: a = 6, b = 2
```

Q2. In an **MBA classroom activity**, teams are anonymized using randomly generated **4-digit Team Codes**. Write a C program that will generate **n** (given by the user) random numbers (between **2500 and 7500**). The program writes each generated team code and the digits of that code in **reverse order** into the file **reverse.txt**.

Project Name: LG01_Q2

File Name: Q2.cpp

Example Run:

```
How many MBA team codes will you generate: 6
6 team codes and their digits in reverse order were written to reverse.txt file
```

reverse.txt

```
Generated number   Digits in reverse order
****************   **********************
            6289    9    8    2    6
            6303    3    0    3    6
            4885    5    8    8    4
            2885    5    8    8    2
            5164    4    6    1    5
            4494    4    9    4    4
```

Q3. In a **car service shop**, a technician enters **positive** integers representing **car part IDs**. Write a C program that reads positive numbers, stores them in an integer array, and finds the count of **prime** and **perfect** part IDs using the functions below.

Write the following function;

- **fillArray**: takes an array as a parameter, reads the positive numbers from the user, stores them in the array, and returns the actual number of elements.
- **isPrime**: takes a number as an input parameter and finds and returns whether the number is prime.
- **isPerfect**: takes a number as an input parameter and finds and returns whether the number is perfect.
- **findPrimePerfect**: takes an array of integers and the array's size as parameters, finds and returns the number of prime and perfect numbers in the given array.

Hints:

- A **Prime number** is a positive integer that is divisible by only 1 and the number itself.
- A **Perfect Number** is defined as a positive integer that is equal to the sum of its positive divisors, excluding the number itself.

Project Name: LG01_Q3

File Name: Q3.cpp

Example Run:

```
Enter a positive car part ID:  6
Enter a positive car part ID:  5
Enter a positive car part ID:  2
Enter a positive car part ID:  65
Enter a positive car part ID:  23
Enter a positive car part ID:  22
Enter a positive car part ID:  11
Enter a positive car part ID:  -6

There are 4 prime numbers
There are 1 perfect numbers
```

```
Enter a positive car part ID:  44
Enter a positive car part ID:  55
Enter a positive car part ID:  1
Enter a positive car part ID:  5
Enter a positive car part ID:  7
Enter a positive car part ID:  9
Enter a positive car part ID:  21
Enter a positive car part ID:  56
Enter a positive car part ID:  -1

There are 2 prime numbers
There are 0 perfect numbers
```

Q4. In a **rock music festival**, each band performs **4 sets**, and the judges give a score for each set. Write a C program that reads **Band IDs** and **set scores** of several rock bands from the file **"rock.txt"**; finds and displays the **average of each set** and the **average of each band** using the functions below.

Write the following functions;

- **readFromFile**: takes a file pointer, a one-dim array to keep the band IDs, and a two-dimensional array to keep the set scores as parameters. The function reads the band IDs into the one-dim array and the 4 set scores into the two-dim array from the specified file. The function also returns the number of bands.
- **findTeamAvg**: takes the two-dim scores array and the number of bands as input parameters, finds the average of each band, and stores the averages into a one-dim array.
- **findGameAvg**: takes the two-dim scores array and the number of bands as input parameters, finds the average of each set, and stores the averages into a one-dim array.
- **displayGameAvg**: takes the one-dim array that keeps the set averages as an input parameter and displays the averages of all sets on the screen.

Project Name: LG01_Q4

File Name: Q4.cpp

Example Run:

```
Band Number  Average
***********  *******
 12          483.50
 24          436.25
 33          505.25
 45          470.00
 57          517.50
 68          449.00
 79          444.25
 89          500.00
 96          484.00
 98          455.50

Set Number  Average
**********  *******
1           475.7
2           482.1
3           496.0
4           444.3
```

rock.txt

```
12 482 570 500 382
24 350 395 575 425
33 475 482 552 512
45 552 545 418 365
57 660 385 475 550
68 446 520 345 485
79 273 582 498 424
89 445 510 570 475
96 624 347 465 500
98 450 485 562 325
```

Additional Questions

AQ1. Write a program that reads numbers from a text file into a two-dimensional integer array. There are 5 columns in each row, while the number of rows is unknown. If the matrix is square, the program will calculate the sum of the elements on the minor diagonal; otherwise, it will calculate the product of elements in the given row.

Write the following functions;

- **display**: displays all elements in the matrix.
- **sumOfRow**: finds the sum of the elements in the specified row of the matrix.
- **productOfMinor**: finds the product of the elements on the minor diagonal of the matrix.

Project Name: LG01_AQ1

File Name: AQ1.cpp

Example Run:

```
   2  52   4   7   8
   3  36  95  47  48
  26  12  25  41  85
  15  36  45  73   5
  48   7  11  98  12
  79  35  64  72  19
  36  44  21  36  15
  45  77   5  25   4
  14   4   6  78  23
  78  36  12   3  98
  25  89   7  12  54
  95  26  36  85  74
  42  78  69  42   1
```

input1.txt

```
2 52 4 7 8
3 36 95 47 48
26 12 25 41 85
15 36 45 73 5
48 7 11 98 12
79 35 64 72 19
36 44 21 36 15
45 77 5 25 4
14 4  6 78 23
78 36 12 3 98
25 89 7 12 54
95 26 36 85 74
42 78 69 42 1
```

AQ2. Write a modular C program that reads the text file named "**numbers.txt**" and stores the numbers in an array using a function named **readFile**() that takes in the file stream and the array as parameters to read the data from the text file into the array. Then, write a function named **menu**() that displays the menu as shown in the example run below to read, validate, and return the user's choice. Depending on this choice, perform the operation mentioned in the menu item selected, and write each menu item as a separate function.

Project Name: LG01_AQ2

File Name: AQ2.cpp

Example Run:

```
DISPLAY
1. All numbers
2. Even numbers
3. Subscripts of odd numbers
4. The numbers with even subscripts
5. Minimum number
6. Subscript of maximum number
7. Exit

Enter your choice: 1

All numbers
************
15 24 65 2 33 78 5 61 4 42 23 1 12 18 32 68 123 111 75

DISPLAY
1. All numbers
2. Even numbers
3. Subscripts of odd numbers
4. The numbers with even subscripts
5. Minimum number
6. Subscript of maximum number
7. Exit

Enter your choice: 2

Even numbers
************
24 2 78 4 42 12 18 32 68

DISPLAY
1. All numbers
2. Even numbers
3. Subscripts of odd numbers
4. The numbers with even subscripts
5. Minimum number
6. Subscript of maximum number
7. Exit

Enter your choice: 3

Subscripts of odd numbers
*************************
0 2 4 6 7 10 11 16 17 18

DISPLAY
1. All numbers
2. Even numbers
3. Subscripts of odd numbers
4. The numbers with even subscripts
5. Minimum number
6. Subscript of maximum number
7. Exit

Enter your choice: 4

The numbers with even subscripts
********************************
15 65 33 5 4 23 12 32 123 75

DISPLAY
1. All numbers
2. Even numbers
3. Subscripts of odd numbers
4. The numbers with even subscripts
5. Minimum number
6. Subscript of maximum number
7. Exit

Enter your choice: 5

Minimum number
***************
1

DISPLAY
1. All numbers
2. Even numbers
3. Subscripts of odd numbers
4. The numbers with even subscripts
5. Minimum number
6. Subscript of maximum number
7. Exit

Enter your choice: 6

Subscript of maximum number
***************************
16

DISPLAY
1. All numbers
2. Even numbers
3. Subscripts of odd numbers
4. The numbers with even subscripts
5. Minimum number
6. Subscript of maximum number
7. Exit

Enter your choice: 7
```