الجمهورية الجزائرية الديمقراطية الشعبية
People's Democratic Republic of Algeria

Ministry of Higher Education and Scientific Research

National School of Cybersecurity

Department of Foundation Training

وزارة التعليم العالي والبحث العلمي

المدرسة الوطنية العليا في الأمن السيبراني

قسم التكوين القاعدي

First-Semester Project: Security Utility Library Suite

Level: 1st Year of the Foundation training **Material:** Algorithms and Static Data Structures

1. General objective

Develop a modular security toolkit using C programming language, composed of several libraries that simulate key cybersecurity concepts.

Students will implement multiple functions and procedures per library and connect them through a main menu (or optional simple interface).

2. Overall structure

No	Library	Theme	Approx. modules
1.	Encryption & Decryption	Classical cryptography	20
2.	Math & Security Tools	Numeric and logic operations	25
3.	User Management	Authentication & account control	20
4.	Audit & Security Analysis	Text and password auditing	20
5.	Log Management & Analysis	Event history and reporting	20
Total			**≈105 modules**

3. Encryption and decryption library

3.1. Proposed structure

```
struct Message{
        char text[200];
        int length;
};
```

3.2. Functions and Procedures

No	Module	Description
1.	void inputMessage(struct Message m[]);	Asks the user to enter a message.
2.	void displayMessage(struct Message m);	Displays the message.
3.	int isUppercase(char c);	Checks if a character is uppercase.
4.	int isLowercase(char c);	Checks if a character is lowercase.
5.	int isAlphabetic(char c);	Tests if a character is a letter.
6.	void toUppercase(struct Message m);	Converts the whole message to uppercase.
7.	void toLowercase(struct Message m);	Converts all letters to lowercase.
8.	void reverseMessage(struct Message m);	Reverses the message.
9.	void removeSpaces(struct Message m);	Removes spaces.
10.	void encryptCesar(struct Message m, int key);	Applies a Caesar cipher.

First-Semester Project: Security Utility Library Suite

Level: 1st Year of the Foundation training **Material:** Algorithms and Static Data Structures

11.	void decryptCesar(struct Message m, int key);	Decrypts a Caesar cipher.
12.	void encryptXOR(struct Message m, int key);	Applies XOR cipher.
13.	void decryptXOR(struct Message m, int key);	Reverses XOR cipher.
14.	void encryptSubstitution(struct Message m, char key[26]);	Encrypts using letter substitution.
15.	void decryptSubstitution(struct Message m, char key[26]);	Decrypts a substitution cipher.
16.	int isValidKey(char key[26]);	Checks if the key is valid.
17.	int compareMessages(struct Message m1, struct Message m2);	Compares two texts.
18.	int countCharacter(struct Message m, char c);	Counts occurrences of a character.
19.	void frequencyAnalysis(struct Message m);	Calculates frequency of each letter.
20.	int coincidenceIndex(struct Message m);	Estimates coincidence index.

4. Mathematical and security tools library

4.1. Structure

```
struct Matrix{
        int data[10][10];
        int n, p;
};
```

4.2. Functions and Procedures

No	Modules	Description
1.	int isEven(int n);	Checks if a number is even.
2.	int isPrime(int n);	Checks if a number is prime.
3.	int gcd(int a, int b);	Computes the greatest common divisor.
4.	int lcm(int a, int b);	Computes the least common multiple.
5.	int modExp(int base, int exp, int mod);	Performs modular exponentiation.
6.	int factorial(int n);	Computes factorial.
7.	int sumDigits(int n);	Sums digits of a number.
8.	int reverseNumber(int n);	Reverses a number.
9.	int isPalindromeNumber(int n);	Checks if a number is palindrome.
10.	int sumDivisors(int n);	Sums divisors of a number.
11.	int isPerfectNumber(int n);	Checks for perfect number.

First-Semester Project: Security Utility Library Suite

Level: 1st Year of the Foundation training **Material:** Algorithms and Static Data Structures

12.	int isArmstrong(int n);	Checks for Armstrong number.
13.	int randomNumber(int min, int max);	Generates random integer.
14.	int sumArray(int T[], int n);	Sums array elements.
15.	float averageArray(int T[], int n);	Computes average.
16.	int maxArray(int T[], int n);	Finds max.
17.	int minArray(int T[], int n);	Finds min.
18.	void sortAscending(int T[], int n);	Sorts array ascending.
19.	void displayMatrix(struct Matrix M);	Prints matrix.
20.	void addMatrices(struct Matrix A, struct Matrix B, struct Matrix C);	Adds two matrices.
21.	void multiplyMatrices(struct Matrix A, struct Matrix B, struct Matrix C);	Multiplies two matrices.
22.	void transposeMatrix(struct Matrix A, struct Matrix T);	Transposes matrix.
23.	int determinant2x2(int A[2][2]);	Computes determinant 2x2.
24.	int isSymmetric(struct Matrix M);	Checks if matrix is symmetric.
25.	int isIdentity(struct Matrix M);	Checks if matrix is identity.

5. User management library

5.1. Structure

```
struct User{
        char name[20];
        char password[20];
        int role;               // 0: user, 1: admin
        int state;              // 0: active, 1: blocked
};
```

5.2. Functions and Procedures

No	Modules	Description
1.	void initUsers(struct User users[], int n);	Initializes user list.
2.	void displayUsers(struct User users[], int n);	Displays all users.
3.	void addUser(struct User users[], int n);	Adds new user.
4.	void deleteUser(struct User users[], int n, char name[]);	Deletes a user.
5.	int searchUser(struct User users[], int n, char name[]);	Searches for a user.
6.	void changePassword(struct User users[], int n, char name[]);	Changes user password.

First-Semester Project: Security Utility Library Suite

Level: 1st Year of the Foundation training **Material:** Algorithms and Static Data Structures

7.	int checkLogin(struct User users[], int n, char name[], char pass[]);	Verifies credentials.
8.	int strongPassword(char pass[]);	Checks password strength.
9.	void blockUser(struct User users[], int n, char name[]);	Blocks a user.
10.	void unblockUser(struct User users[], int n, char name[]);	Unblocks a user.
11.	void changeRole(struct User users[], int n, char name[], int role);	Changes role.
12.	void listAdmins(struct User users[], int n);	Lists administrators.
13.	int stringLength(char str[]);	Returns string length.
14.	int containsUppercase(char str[]);	Checks for uppercase.
15.	int containsLowercase(char str[]);	Checks for lowercase.
16.	int containsDigit(char str[]);	Checks for digits.
17.	int containsSymbol(char str[]);	Checks for special characters.
18.	void userStatistics(struct User users[], int n);	Displays statistics.
19.	void saveUsers(struct User users[], int n);	Saves users to file.
20.	void loadUsers(struct User users[], int n);	Loads users from file.

6. Security audit and analysis library

6.1. Functions and Procedures

No	Modules	Description
1.	int countUppercase(char text[]);	Counts uppercase letters.
2.	int countLowercase(char text[]);	Counts lowercase letters.
3.	int countDigits(char text[]);	Counts digits.
4.	float percentUppercase(char text[]);	Calculates percentage of uppercase.
5.	int textLength(char text[]);	Returns length of text.
6.	void displayTextStats(char text[]);	Displays general statistics.
7.	int veryStrongPassword(char pass[]);	Checks if password is very strong.
8.	void generateKey(int length, char key[]);	Generates random key.
9.	int isHexKey(char key[]);	Verifies hexadecimal format.
10.	void generateRandomPassword(int length, char pass[]);	Generates random password.
11.	int passwordScore(char pass[]);	Returns password strength score.
12.	float averageScore(struct User users[], int n);	Computes average score.
13.	void displaySecurityReport(struct User users[], int n);	Displays global report.
14.	int countStrongUsers(struct User users[], int n);	Counts users with strong passwords.
15.	void showSecurityTips();	Prints general security advice.

First-Semester Project: Security Utility Library Suite

Level: 1st Year of the Foundation training **Material:** Algorithms and Static Data Structures

16.	int checkEmailFormat(char email[]);	Verifies valid email format.
17.	int checkLoginFormat(char name[]);	Verifies login validity.
18.	void generateHexKey(int length, char key[]);	Generates hexadecimal key.
19.	void top3Passwords(struct User users[], int n);	Displays top 3 passwords.
20.	int globalSecurityLevel(struct User users[], int n);	Computes global level.

7. Log management and analysis library

7.1. Structure

```
struct Log{
        char user[20];
        char action[50];
        char date[20];
        char time[10];
        int code;           // 0 info, 1 warning, 2 error
};
```

7.2. Functions and Procedures

No	Modules	Description
1.	void initLogs(struct Log logs[], int n);	Initializes log list.
2.	void addLog(struct Log logs[], int n, char user[], char action[], int code);	Adds a log entry.
3.	void displayLogs(struct Log logs[], int n);	Displays all logs.
4.	void searchLogsByUser(struct Log logs[], int n, char user[]);	Searches logs by user.
5.	void searchLogsByDate(struct Log logs[], int n, char date[]);	Searches logs by date.
6.	int countErrorLogs(struct Log logs[], int n);	Counts error entries.
7.	int countLoginLogs(struct Log logs[], int n);	Counts login events.
8.	int countBlockedLogs(struct Log logs[], int n);	Counts blocked attempts.
9.	void displayLogStats(struct Log logs[], int n);	Shows statistics.
10.	void sortLogsByDate(struct Log logs[], int n);	Sorts logs by date.
11.	void sortLogsByUser(struct Log logs[], int n);	Sorts by username.
12.	int detectSuspiciousActivity(struct Log logs[], int n, char user[]);	Detects anomalies.
13.	int dailyConnections(struct Log logs[], int n, char date[]);	Counts daily connections.
14.	float errorRate(struct Log logs[], int n);	Computes error percentage.

First-Semester Project: Security Utility Library Suite

Level: 1st Year of the Foundation training **Material:** Algorithms and Static Data Structures

15.	void exportLogsCSV(struct Log logs[], int n);	Exports logs to CSV.
16.	void importLogsCSV(struct Log logs[], int n);	Imports logs from CSV.
17.	void clearLogs(struct Log logs[], int n);	Clears all logs.
18.	void recentLogs(struct Log logs[], int n, int nb);	Displays last events.
19.	void archiveLogs(struct Log logs[], int n);	Archives old logs.
20.	void showTopErrors(struct Log logs[], int n);	Displays top frequent errors.

8. Main menu

```
===============================================
          SECURITY UTILITY LIBRARY SYSTEM
===============================================
1. Encryption and Decryption Library
2. Mathematical and Security Tools
3. User Management System
4. Security Audit and Analysis
5. Log Management and Monitoring
6. Help
7. About
0. Exit
-----------------------------------------------
Enter your choice: __
```

9. Optional part: User Interface (UI)

9.1. Objective

Create a simple text or graphical interface to navigate between the libraries and functions.

9.2. Suggested libraries

Library	Type	Description
ncurses	Console	Interactive text menus with colors.
raylib	Graphical	2D windowed menu interface.
GTK	GUI	Full graphical interface (optional advanced bonus).

10. Project deliverables

Each group of students (2 students max) must submit two deliverables:

10.1. Source code folder

A well-organized folder containing:

- One .c file per library (encryption.c, math_tools.c, etc.).

First-Semester Project: Security Utility Library Suite

Level: 1st Year of the Foundation training **Material:** Algorithms and Static Data Structures

- A corresponding .h header file for each module.
- A main program (main.c) that allows the user to test all features via a menu.
- Clear comments for all modules (description, inputs, outputs).

10.2. Analysis report (PDF document)

Each student (or team) must also submit a PDF report (recommended: 10 pages) named **SecurityLibrary_Analysis_<GroupName>.pdf**

- **Cover page**: Project title, names of students, group, and date.
- **Introduction**: Short overview of the objective and organization of the project.
- **Module analysis**: chose two modules from each section and propose a detailed analysis for them.
- **Bonus work (if implemented)**: Description of the user interface or additional features.
- **Conclusion**: Lessons learned, difficulties faced, and possible improvements.

NB:

- The report must be clear, structured, and typed (PDF only).
- Handwritten submissions are not accepted.