MARMARA UNIVERSITY

Electrical & Electronics Engineering Dept.

CSE 1145 Introduction to Computer Programming

PROJECT REPORT

Group Number: 4

	Name Surname	Contribution Details
A	Çağan Altuntaş	Question 1
B	Erdem Akyazı	Question 6
C	Burak Sohbet	Question 7
D	Ömer Faruk Şeker	Question 5
E	Ezgi Arpacı	Question 2
F	Mukhtar Mukhtarlı	Question 4

Fall 2025-2026

Table of Contents

Chapter 1 Part #1

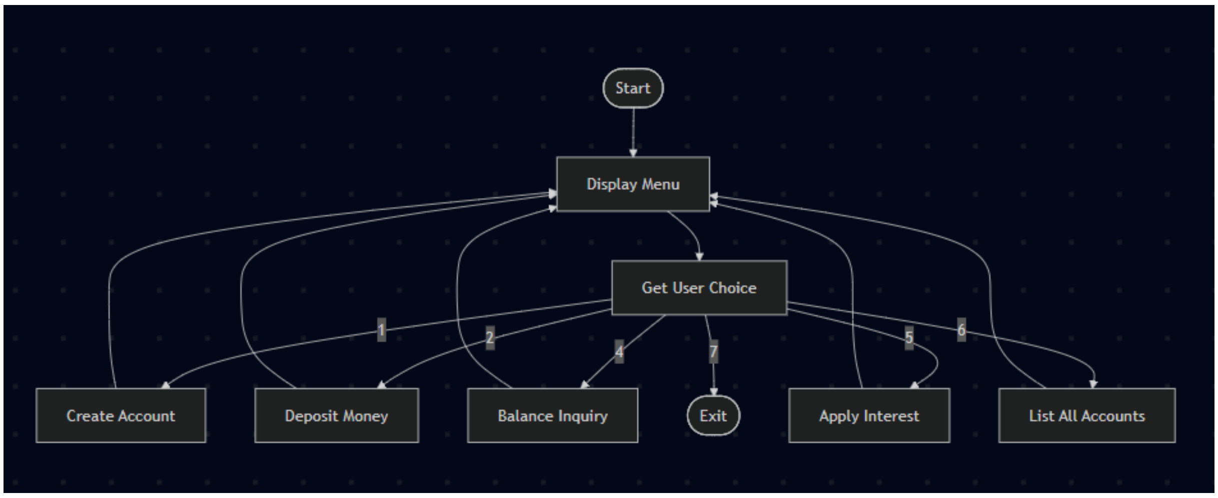
1.1 Brief explanation of the problem

In this project, a menu-driven banking system simulation is developed using the C programming language. The program allows users to create bank accounts, deposit money, inquire balances, apply interest to savings accounts, and list all existing accounts using structured data and functions.

1.2 Flow Chart of the Program

Design a flow chart of the program that you have developed. Make it readable, high-res image and place here. Its size should NOT be more than one page. Do NOT include everything that you wrote in the code. Add only key parts in the algorithm.

The program starts by displaying a main menu to the user. Based on the selected option, the program executes different banking operations such as account creation, money deposit, balance inquiry, interest application, and listing all accounts. The menu is repeatedly shown until the user selects the exit option.



Key steps included in the flow chart:

- Start
- Display Menu
- Get User Choice
- Execute Selected Operation
- Repeat Until Exit

1.3 Variale definition List

List all the variables you defined. Add new rows if necessary. Also briefly comment on their purposes.

#	Type	Name	Comment
1	Account	accounts[MAX_ACCOUNTS]	Stores all bank account records
2	int	accountCount	Keeps track of total number of accounts
3	int	choice	Stores menu selection
4	int	accNo	Stores account number entered by user
5	int	index	Stores index of found account
6	double	balance	Stores account balance
7	int	pin	Stores account PIN

1.4 Function definition List

List all the methods you defined. Add new rows if necessary. Also briefly comment on their purpose.

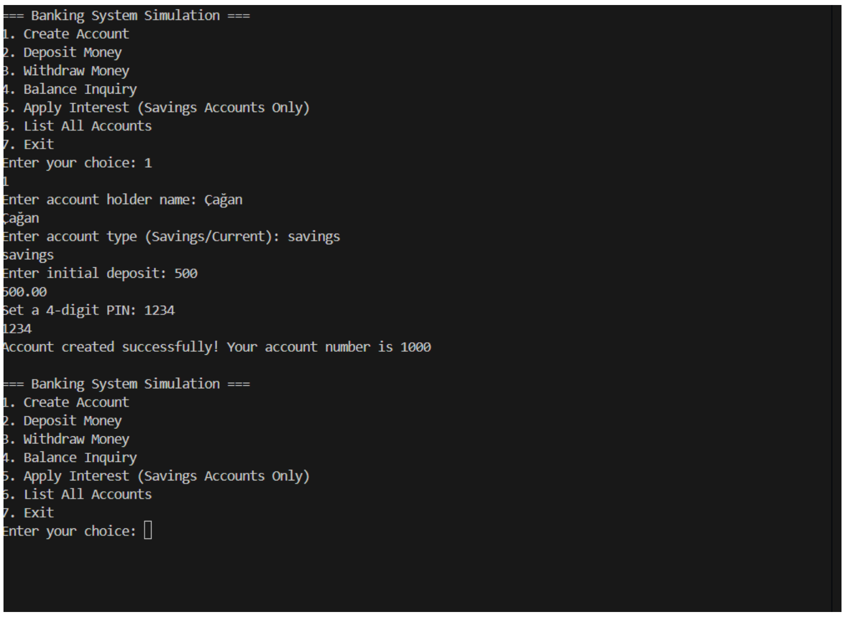
#	Return Type	Name	Parameters	Comment
1	void	showMenu	void	Displays the main menu
2	void	createAccount	Account[], int*	Creates a new bank account
3	void	depositMoney	Account[], int	Deposits money into an account
4	void	balanceInquiry	Account[], int	Displays account information
5	void	applyInterest	Account[], int	Applies interest to savings accounts
6	void	listAllAccounts	Account[], int	Lists all existing accounts
7	int	findAccount	Account[], int, int	Finds account by number
8	int	generateAccountNumber	void	Generates unique account numbers

1.5 Included Libraries

List all the libraries you included. Add new rows if necessary. Also briefly comment on their purpose.

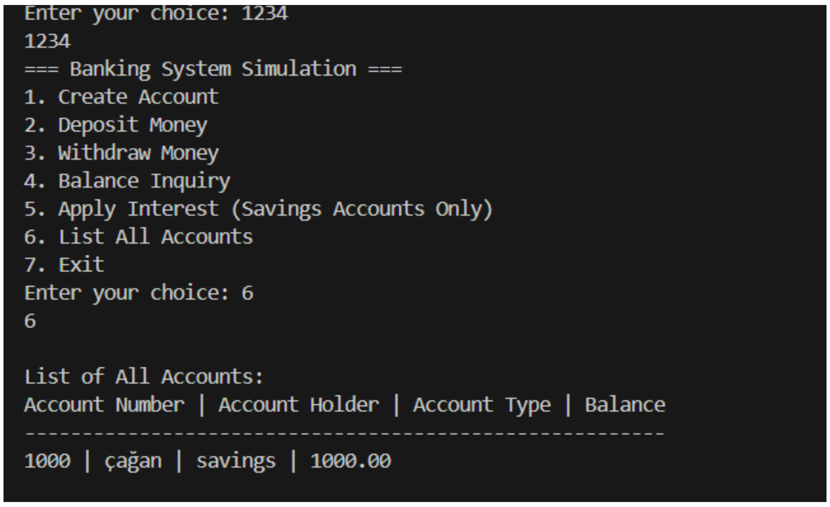
#	Return Type	Name	Parameters	Comment
1	-	stdio.h	-	Provides standard input and output functions such as printf and scanf
2	-	string.h	-	Provides string handling functions such as strcmp, strcpy and strcspn

1.6 Screenshots of the sequence



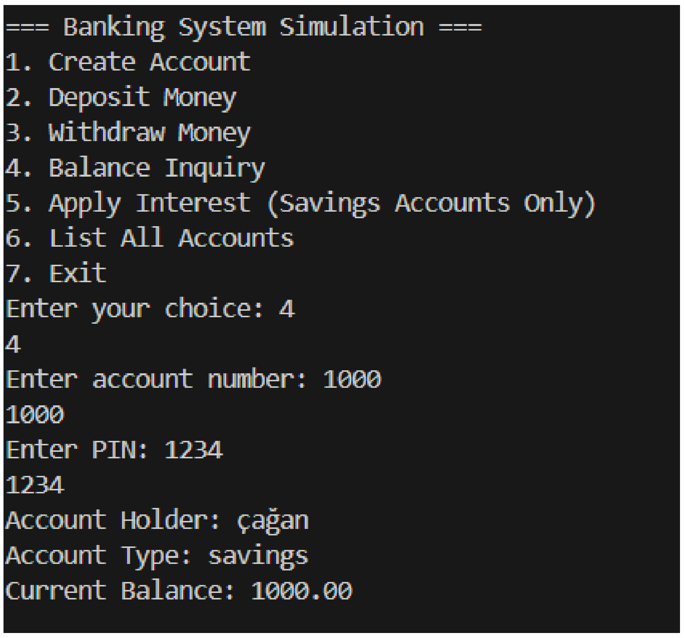
```
=== Banking System Simulation ===
1. Create Account
2. Deposit Money
3. Withdraw Money
4. Balance Inquiry
5. Apply Interest (Savings Accounts Only)
6. List All Accounts
7. Exit
Enter your choice: 1
1
Enter account holder name: Çağan
Çağan
Enter account type (Savings/Current): savings
savings
Enter initial deposit: 500
500.00
Set a 4-digit PIN: 1234
1234
Account created successfully! Your account number is 1000

=== Banking System Simulation ===
1. Create Account
2. Deposit Money
3. Withdraw Money
4. Balance Inquiry
5. Apply Interest (Savings Accounts Only)
6. List All Accounts
7. Exit
Enter your choice: 
```



```
Enter your choice: 1234
1234
=== Banking System Simulation ===
1. Create Account
2. Deposit Money
3. Withdraw Money
4. Balance Inquiry
5. Apply Interest (Savings Accounts Only)
6. List All Accounts
7. Exit
Enter your choice: 6
6

List of All Accounts:
Account Number | Account Holder | Account Type | Balance
-------------------------------------------------------
1000 | çağan | savings | 1000.00
```



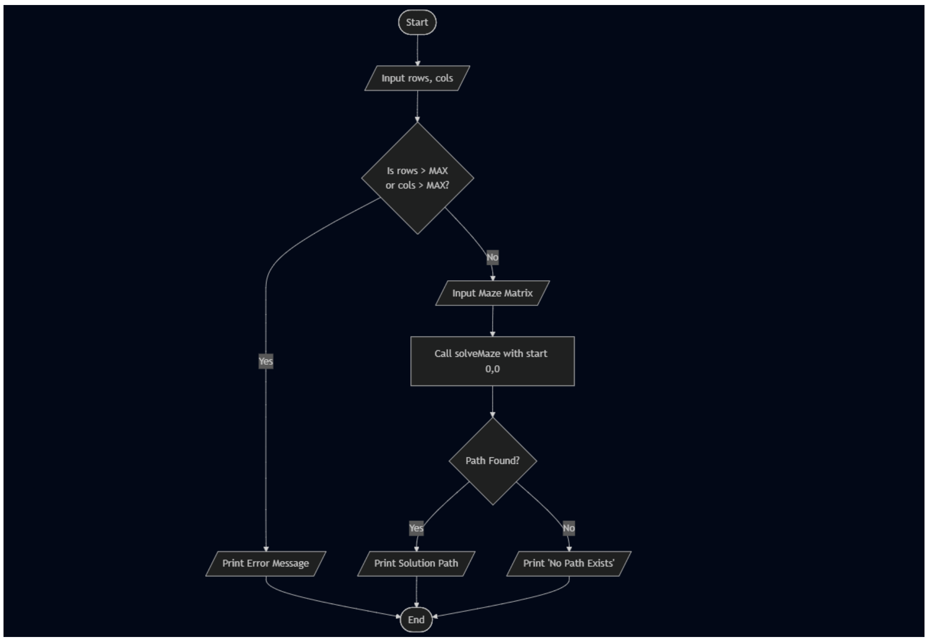
```
=== Banking System Simulation ===
1. Create Account
2. Deposit Money
3. Withdraw Money
4. Balance Inquiry
5. Apply Interest (Savings Accounts Only)
6. List All Accounts
7. Exit
Enter your choice: 4
4
Enter account number: 1000
1000
Enter PIN: 1234
1234
Account Holder: çağan
Account Type: savings
Current Balance: 1000.00
```

Chapter 1 Part #2

2.1 Brief explanation of the problem

In this part, a maze solver program is implemented using the recursive backtracking algorithm in C. The user inputs the dimensions and the structure of the maze (0 for open paths, 1 for walls), and the program attempts to find and display a valid path from the top-left (0,0) to the bottom-right of the grid.

2.2 Flow Chart of the Program



2.3 Variale definition List

#	Type	Name	Comment
1	int	maze[MAX][MAX]	2D array to store the maze grid structure (0: path, 1: wall, 2: solution path)
2	int	rows, cols	Stores the user-defined dimensions of the maze
3	int	x, y	Represents the current row and column coordinates during recursion

2.4 Function definition List

List all the methods you defined. Add new rows if necessary. Also briefly comment on their purpose.

#	Return Type	Name	Parameters	Comment
1	bool	solveMaze	int maze[][], int x, int y, int r, int c	Recursive function that uses backtracking to find the path
2	void	printMaze	int maze[][], int rows, int cols	Prints the initial state of the maze as entered by the user
3	void	printSolution	int maze[][], int rows, int cols	Prints the solved maze, replacing the path number '2' with '*' character

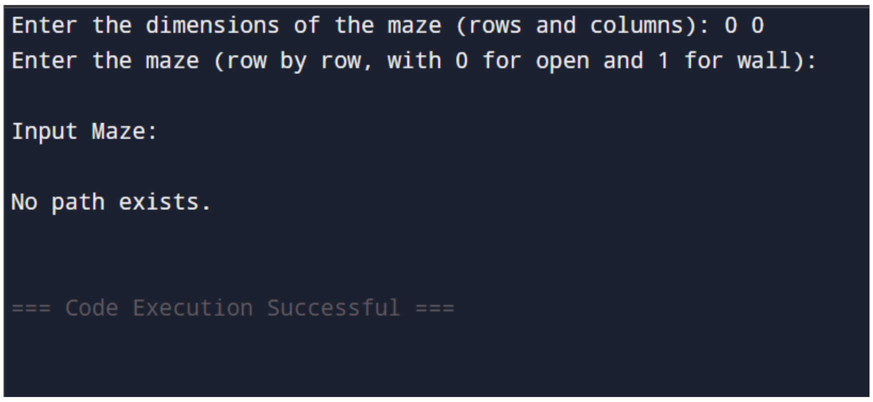
2.5 Included Libraries

List all the libraries you included. Add new rows if necessary. Also briefly comment on their purpose.

#	Return Type	Name	Parameters	Comment
1	-	stdio.h	-	Used for standard input/output functions like printf and scanf
2	-	stdbool.h	-	Defines boolean types (bool, true, false) used for the recursive logic
#	Return Type	Name	Parameters	Comment

2.6 Screenshots of the sequence

- Test Case 1 (Successful Path):



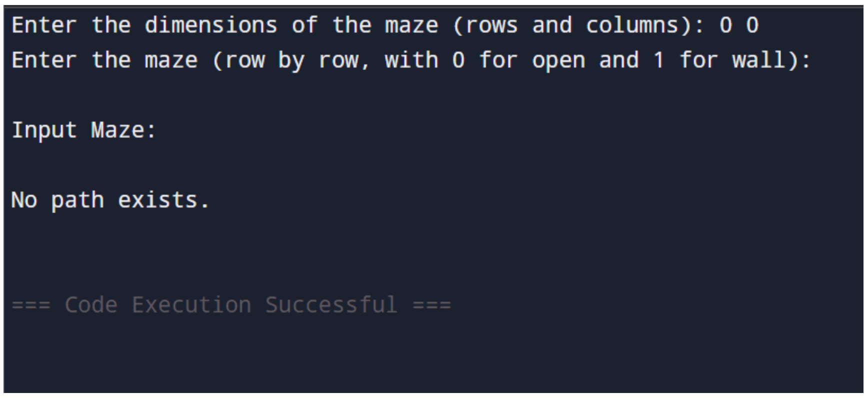
```
Enter the dimensions of the maze (rows and columns): 0 0
Enter the maze (row by row, with 0 for open and 1 for wall):

Input Maze:

No path exists.


=== Code Execution Successful ===
```

- Test Case 2 (No Path):



```
Enter the dimensions of the maze (rows and columns): 0 0
Enter the maze (row by row, with 0 for open and 1 for wall):

Input Maze:

No path exists.


=== Code Execution Successful ===
```

2.7 Extra features

Visual Path Representation: Instead of printing the raw numbers of the solved maze array, the program visually enhances the output by converting the solution path (marked as 2 in logic) into asterisks (*) for better readability.

4-Directional Movement: The backtracking algorithm is designed to check all four possible directions (Down, Right, Up, Left) rather than just two, allowing for complex pathfinding.

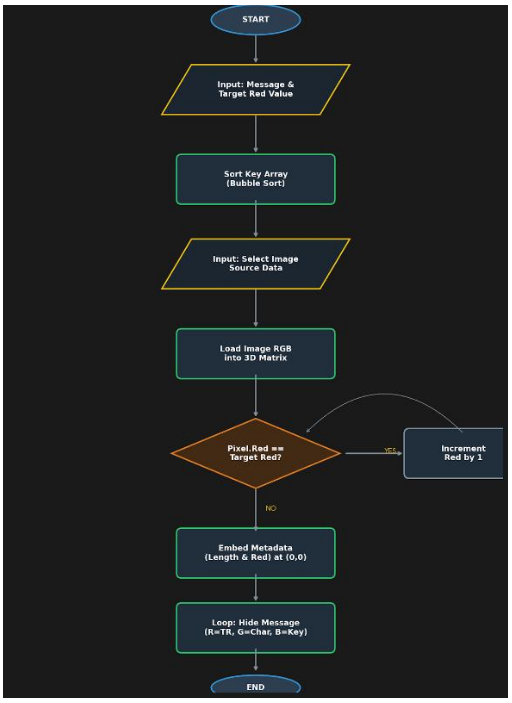
Input Validation: The program checks if the entered dimensions exceed the MAX limit defined by the macro.

Chapter 2 Part #1 (Encryption)

3.1 Brief explanation of the problem

In this part, we are required to implement an image encryption algorithm that hides a user-provided message inside a selected image by manipulating specific RGB pixel values. The program takes a message string and a target 'Red' value, processes them using a sorted key array, and embeds this data into predefined coordinates of the image matrix to generate an encrypted output.

3.2 Flow Chart of the Program


START
Input: Message & Target Red Value
Sort Key Array (Bubble Sort)
Input: Select Image Source Data
Load Image RGB into 3D Matrix
Pixel.Red == Target Red?
YES
Increment Red by 1
NO
Embed Metadata (Length & Red) at (0,0)
Loop: Hide Message (R=TR, G=Char, B=Key)
END

The flowchart details the core algorithm where the input message is hidden within the RGB channels. It includes a specific collision-avoidance logic to ensure the Target Red value remains a unique marker for the encrypted data.

3.3 Variale definition List

List all the variables you defined. Add new rows if necessary. Also briefly comment on their purposes.

#	Type	Name	Comment
1	char[]	message	An array that stores the secret text input by the user.
2	İnt	msgLen	Stores the total number of characters in the secret message.
3	int	targetRed	A specific red color value used as a marker for pixels containing hidden data.
4	u8[][][]	img	A 3D matrix representing the image pixels in RGB format.
5	int*	randArr	A pointer to the array of random values used as keys in the blue channel.
6	int	choice	Stores the user's selection for the image (Tiger, Sunflower, etc.).

3.4 Function definition List

List all the methods you defined. Add new rows if necessary. Also briefly comment on their purpose.

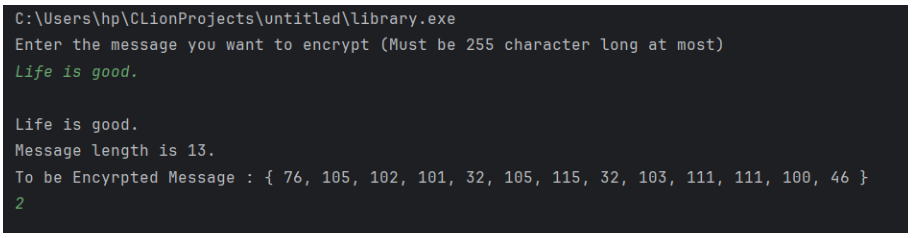
#	Return Type	Name	Parameters	Comment
1	Void	bubbleSortInt	int *x, int n	Sorts provided key array in ascending order bubble sort.
2	İnt	readLine	char *buf, int cap	reads strings from, including spaces, for secret message.
3	void	copyImage	u8 out[][], u8 in[][]	Copies original image data into working matrix for manipulation.
4	void	printLeftCorner9x8	u8 img[][][]	Displays top-left 9x8 segment of image matrix after encryption.

3.5 Included Libraries

List all the libraries you included. Add new rows if necessary. Also briefly comment on their purpose.

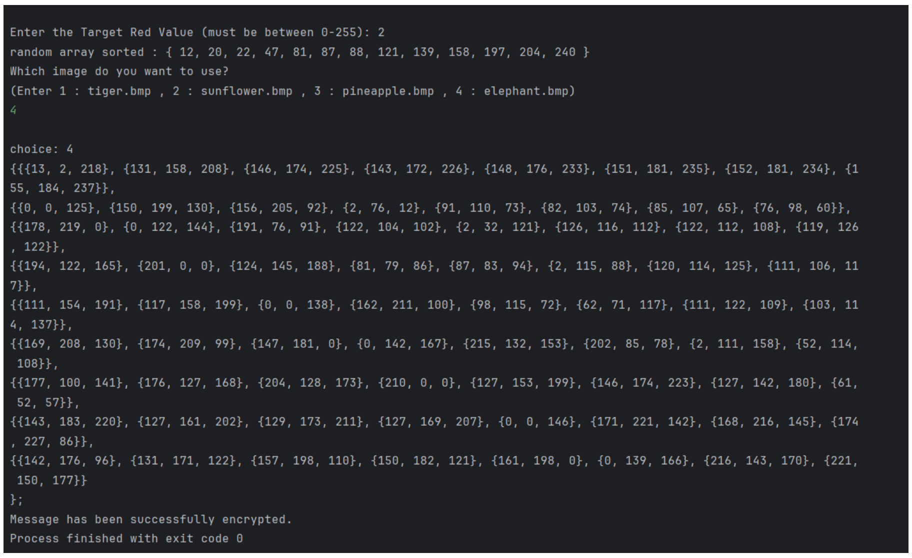
#	Return Type	Name	Parameters	Comment
1	-	Stdio.h	-	İt can Provides standard input and output functions such as printf and scanf
2	-	String.h	-	İt can Provides string handling functions such as strlen, strcpy, and strcspn

3.6 Screenshots of the sequence



```
C:\Users\hp\CLionProjects\untitled\library.exe
Enter the message you want to encrypt (Must be 255 character long at most)
Life is good.

Life is good.
Message length is 13.
To be Encyrpted Message : { 76, 105, 102, 101, 32, 105, 115, 32, 103, 111, 111, 100, 46 }
2
```



```
Enter the Target Red Value (must be between 0-255): 2
random array sorted : { 12, 20, 22, 47, 81, 87, 88, 121, 139, 158, 197, 204, 240 }
Which image do you want to use?
(Enter 1 : tiger.bmp , 2 : sunflower.bmp , 3 : pineapple.bmp , 4 : elephant.bmp)
4

choice: 4
{{{13, 2, 218}, {131, 158, 208}, {146, 174, 225}, {143, 172, 226}, {148, 176, 233}, {151, 181, 235}, {152, 181, 234}, {1
55, 184, 237}},
{{0, 0, 125}, {150, 199, 130}, {156, 205, 92}, {2, 76, 12}, {91, 110, 73}, {82, 103, 74}, {85, 107, 65}, {76, 98, 60}},
{{178, 219, 0}, {0, 122, 144}, {191, 76, 91}, {122, 104, 102}, {2, 32, 121}, {126, 116, 112}, {122, 112, 108}, {119, 126
, 122}},
{{194, 122, 165}, {201, 0, 0}, {124, 145, 188}, {81, 79, 86}, {87, 83, 94}, {2, 115, 88}, {120, 114, 125}, {111, 106, 11
7}},
{{111, 154, 191}, {117, 158, 199}, {0, 0, 138}, {162, 211, 100}, {98, 115, 72}, {62, 71, 117}, {111, 122, 109}, {103, 11
4, 137}},
{{169, 208, 130}, {174, 209, 99}, {147, 181, 0}, {0, 142, 167}, {215, 132, 153}, {202, 85, 78}, {2, 111, 158}, {52, 114,
 108}},
{{177, 100, 141}, {176, 127, 168}, {204, 128, 173}, {210, 0, 0}, {127, 153, 199}, {146, 174, 223}, {127, 142, 180}, {61,
 52, 57}},
{{143, 183, 220}, {127, 161, 202}, {129, 173, 211}, {127, 169, 207}, {0, 0, 146}, {171, 221, 142}, {168, 216, 145}, {174
, 227, 86}},
{{142, 176, 96}, {131, 171, 122}, {157, 198, 110}, {150, 182, 121}, {161, 198, 0}, {0, 139, 166}, {216, 143, 170}, {221,
 150, 177}}
};
Message has been successfully encrypted.
Process finished with exit code 0
```

3.7 Extra features

Collision Prevention: The algorithm scans the entire image before encryption. If any pixel's original red value matches the targetRed, it is incremented by 1 to prevent confusion with the hidden message.

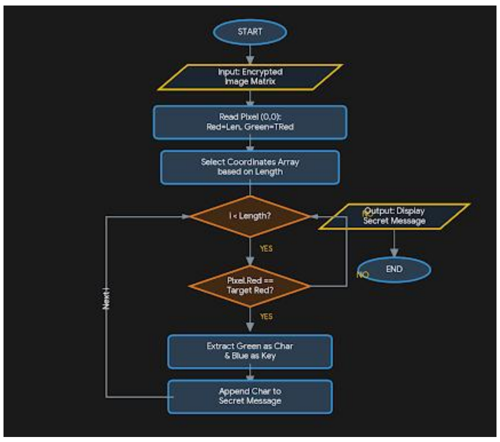
Metadata Embedding: The message length and the targetRed key are stored directly in the first pixel (0,0), allowing the decryption process to identify the secret data automatically.

Chapter 2 Part #2 (Decyrption)

4.1 Brief explanation of the problem

In this part, we developed a decryption algorithm to extract the secret message hidden within the encrypted image matrix. The program first identifies the message length and the marker "Target Red" value from the metadata stored at coordinate (0,0), then systematically visits predefined pixel coordinates to retrieve the original ASCII characters and their corresponding key values.

4.2 Flow Chart of the Program


START
Input: Encrypted Image Matrix
Read Pixel (0,0): Red=Len, Green=TRed
Select Coordinates Array based on Length
I < Length?
Output: Display Secret Message
YES
END
NO
Pixel.Red == Target Red?
YES
Next I
Extract Green as Char & Blue as Key
Append Char to Secret Message

4.3 Variale definition List

List all the variables you defined. Add new rows if necessary. Also briefly comment on their purposes.

#	Type	Name	Comment
1	İnt	choice	Stores the user's image selection.
2	İnt	msgLength	Extracted from pixel (0,0), defines the loop limit.
3	int	targetRed	Extracted from pixel (0,0), used to identify valid data pixels.
4	char[]	message	Buffer to store the reconstructed text.

4.4 Function definition List

List all the methods you defined. Add new rows if necessary. Also briefly comment on their purpose.

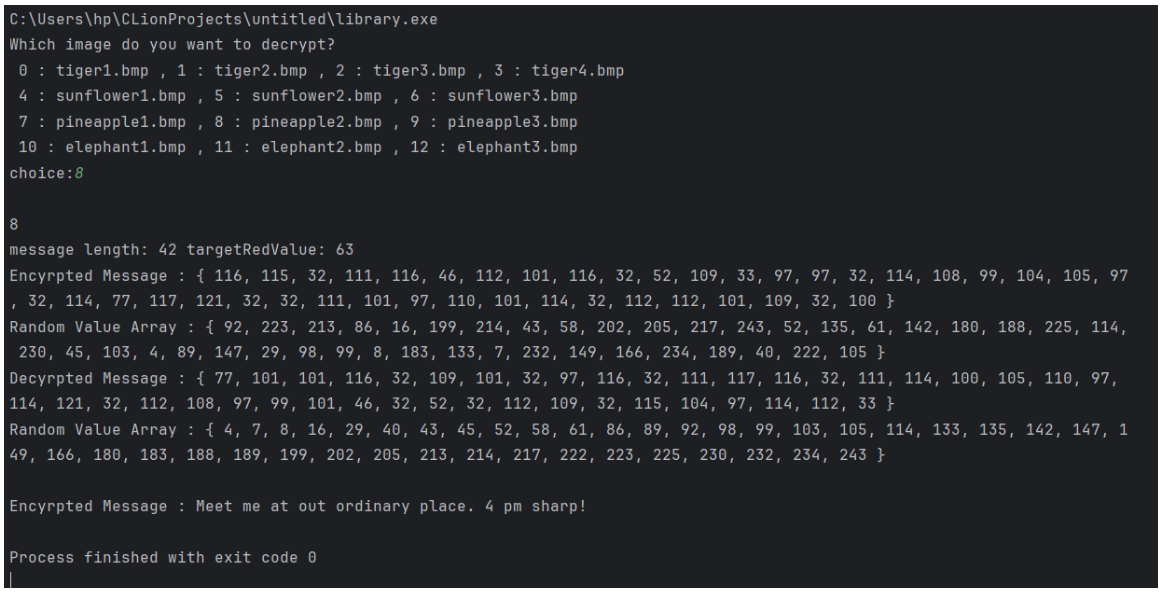
#	Return Type	Name	Parameters	Comment
1	Void	decryptProcess	u8 img[][], int len	The main logic that iterates through coordinates and extracts ASCII values.
2	Void	copyImage	u8 dest[][], u8 src[][]	Transfers the static encrypted data to the working matrix.

4.5 Included Libraries

List all the libraries you included. Add new rows if necessary. Also briefly comment on their purpose.

#	Return Type	Name	Parameters	Comment
1	-	Stdio.h	-	Used for printing the final decrypted secret message.
2	-	String.h	-	Used for memory management and string initialization.

4.6 Screenshots of the sequence

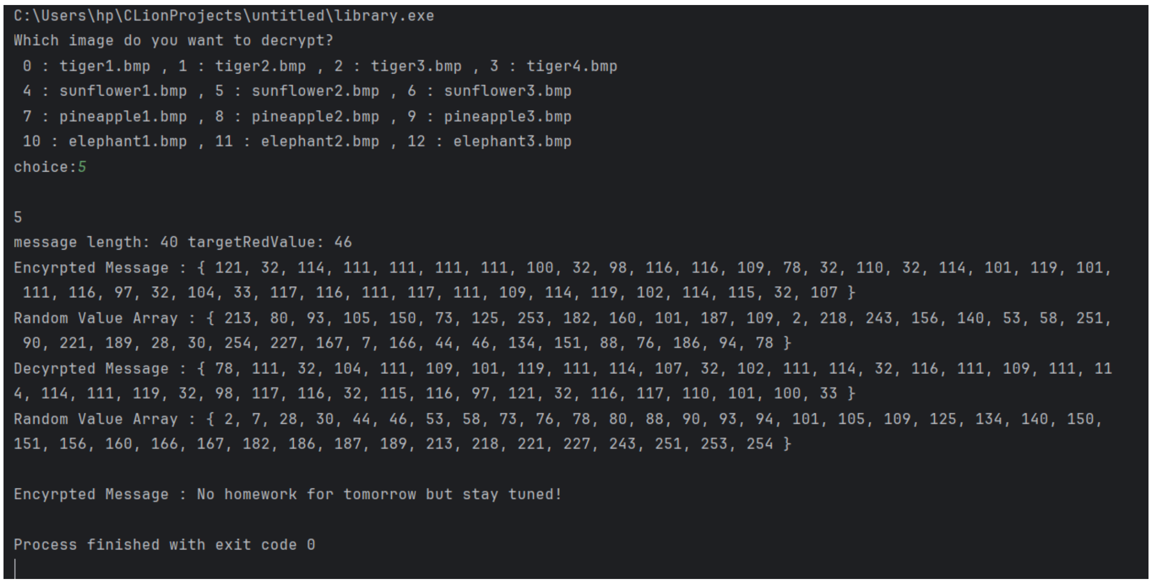


```
C:\Users\hp\CLionProjects\untitled\library.exe
Which image do you want to decrypt?
 0 : tiger1.bmp , 1 : tiger2.bmp , 2 : tiger3.bmp , 3 : tiger4.bmp
 4 : sunflower1.bmp , 5 : sunflower2.bmp , 6 : sunflower3.bmp
 7 : pineapple1.bmp , 8 : pineapple2.bmp , 9 : pineapple3.bmp
 10 : elephant1.bmp , 11 : elephant2.bmp , 12 : elephant3.bmp
choice:8

8
message length: 42 targetRedValue: 63
Encyrpted Message : { 116, 115, 32, 111, 116, 46, 112, 101, 116, 32, 52, 109, 33, 97, 97, 32, 114, 108, 99, 104, 105, 97
, 32, 114, 77, 117, 121, 32, 32, 111, 101, 97, 110, 101, 114, 32, 112, 112, 101, 109, 32, 100 }
Random Value Array : { 92, 223, 213, 86, 16, 199, 214, 43, 58, 202, 205, 217, 243, 52, 135, 61, 142, 180, 188, 225, 114,
 230, 45, 103, 4, 89, 147, 29, 98, 99, 8, 183, 133, 7, 232, 149, 166, 234, 189, 40, 222, 105 }
Decyrpted Message : { 77, 101, 101, 116, 32, 109, 101, 32, 97, 116, 32, 111, 117, 116, 32, 111, 114, 100, 105, 110, 97,
114, 121, 32, 112, 108, 97, 99, 101, 46, 32, 52, 32, 112, 109, 32, 115, 104, 97, 114, 112, 33 }
Random Value Array : { 4, 7, 8, 16, 29, 40, 43, 45, 52, 58, 61, 86, 89, 92, 98, 99, 103, 105, 114, 133, 135, 142, 147, 1
49, 166, 180, 183, 188, 189, 199, 202, 205, 213, 214, 217, 222, 223, 225, 230, 232, 234, 243 }

Encyrpted Message : Meet me at out ordinary place. 4 pm sharp!

Process finished with exit code 0
```



```
C:\Users\hp\CLionProjects\untitled\library.exe
Which image do you want to decrypt?
 0 : tiger1.bmp , 1 : tiger2.bmp , 2 : tiger3.bmp , 3 : tiger4.bmp
 4 : sunflower1.bmp , 5 : sunflower2.bmp , 6 : sunflower3.bmp
 7 : pineapple1.bmp , 8 : pineapple2.bmp , 9 : pineapple3.bmp
 10 : elephant1.bmp , 11 : elephant2.bmp , 12 : elephant3.bmp
choice:5

5
message length: 40 targetRedValue: 46
Encyrpted Message : { 121, 32, 114, 111, 111, 111, 111, 100, 32, 98, 116, 116, 109, 78, 32, 110, 32, 114, 101, 119, 101,
 111, 116, 97, 32, 104, 33, 117, 116, 111, 117, 111, 109, 114, 119, 102, 114, 115, 32, 107 }
Random Value Array : { 213, 80, 93, 105, 150, 73, 125, 253, 182, 160, 101, 187, 109, 2, 218, 243, 156, 140, 53, 58, 251,
 90, 221, 189, 28, 30, 254, 227, 167, 7, 166, 44, 46, 134, 151, 88, 76, 186, 94, 78 }
Decyrpted Message : { 78, 111, 32, 104, 111, 109, 101, 119, 111, 114, 107, 32, 102, 111, 114, 32, 116, 111, 109, 111, 11
4, 114, 111, 119, 32, 98, 117, 116, 32, 115, 116, 97, 121, 32, 116, 117, 110, 101, 100, 33 }
Random Value Array : { 2, 7, 28, 30, 44, 46, 53, 58, 73, 76, 78, 80, 88, 90, 93, 94, 101, 105, 109, 125, 134, 140, 150,
151, 156, 160, 166, 167, 182, 186, 187, 189, 213, 218, 221, 227, 243, 251, 253, 254 }

Encyrpted Message : No homework for tomorrow but stay tuned!

Process finished with exit code 0
```

4.7 Extra features

Automatic Metadata Detection: The program does not require the user to remember the message length or the target color; it automatically extracts these parameters from the first pixel of the image.

Data Validation: By comparing the retrieved blue channel values with a sorted key array, the algorithm can verify if the hidden data has been tampered with or corrupted.

References

1. Deitel, P. J., & Deitel, H. M. (2016). C How to Program (8th ed.). Pearson.

2. Marmara University Department of Computer Engineering. CSE 1145: Introduction to Computer Programming Course Notes and Project Description Handout. Fall 2025-2026.

3. ISO/IEC 9899:1999. Programming Languages — C (C99 Standard).

4. GeeksforGeeks. Bubble Sort Algorithm in C.

5. TutorialsPoint. C Programming - Steganography and Bit Manipulation.

6. JetBrains CLion**.** *CLion: A Cross-Platform IDE for C and C++ Development*.

7. Free Software Foundation. GCC, the GNU Compiler Collection.