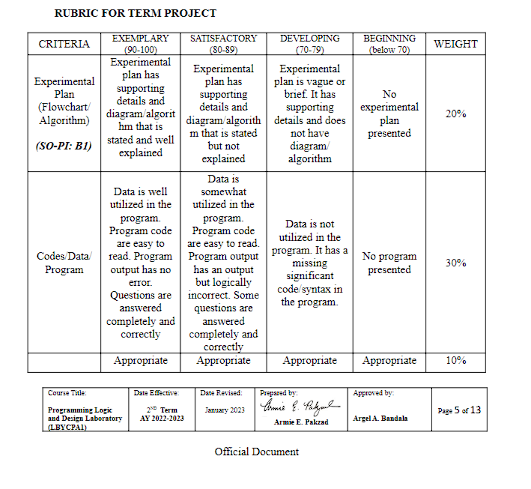
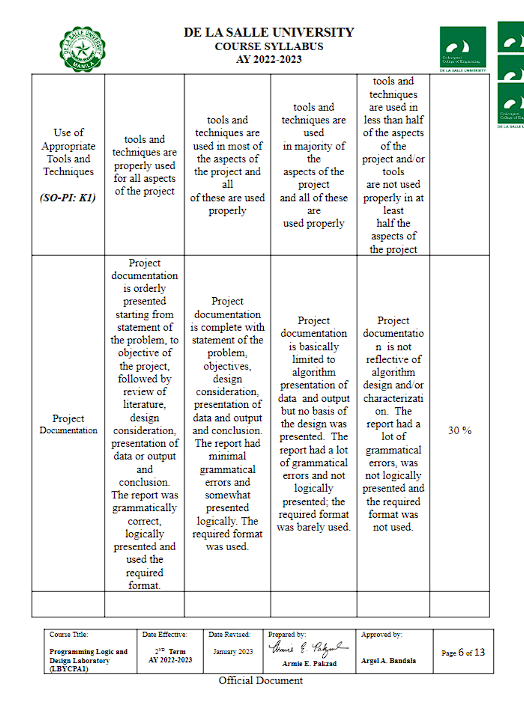
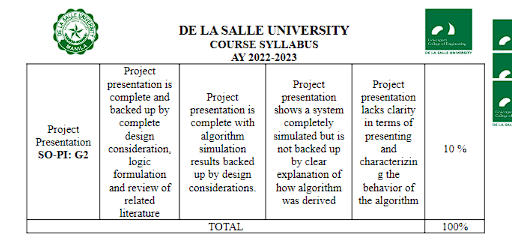
****

****

****

**DE LA SALLE UNIVERSITY - MANILA**

CRYPHYTONOLOGY: An Integration of Python Programming in Message Encryption and Decryption

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

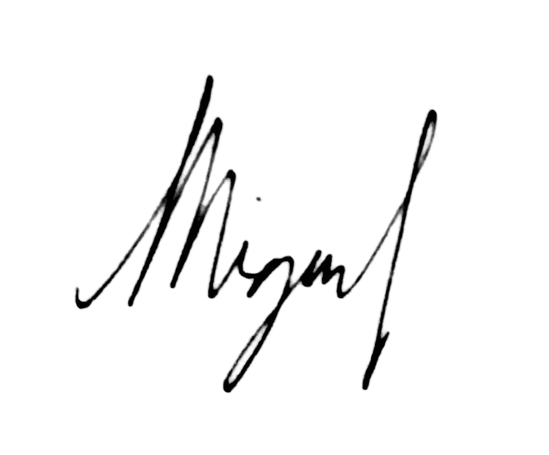
A Term Project

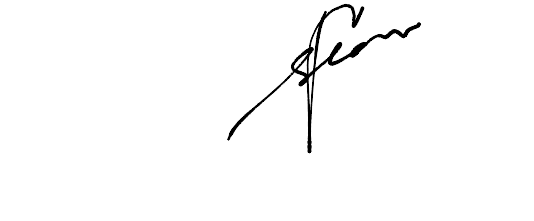
Presented to Mr. Ramon Stephen L. Ruiz

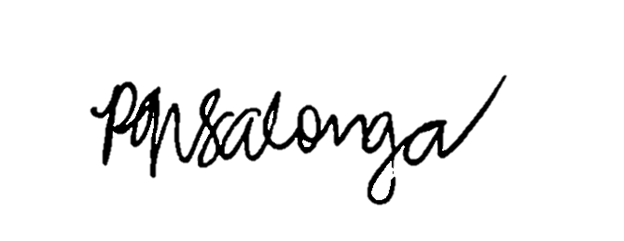
In Partial Fulfillment of the

Requirements for the Course Programming Logic and Design (PROLOGI)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

by

MARTINEZ, Miguel James R. - 

MORALES, Sean Alexander M. - 

SALONGA, Princess Nicole G. -

EQ4

Monday and Thursday (7:30 AM - 10:30 AM)

April 2023

**Table of Contents**

[**I. Introduction 7**](#_f7mzbb6aycj)

[A. Background of the Study 7](#_ks9pb6kie52y)

[B. Problem Statement 8](#_vsmr6cc9tvuj)

[C. Objectives 8](#_wmxwgptkhebu)

[D. Significance of the Project 9](#_noc2mzmsamc4)

[**II. Review of Related Literature 10**](#_8pxz0sigt5q8)

[A. Conceptualization of Cryptography and Cryptosystems 10](#_4pbvb78zq88c)

[Shift Cipher 10](#_di2z9r1wuoty)

[Vigenère Cipher 11](#_d3271bsi1xdo)

[RSA Cryptosystem 12](#_bnnbajov0u4p)

[B. Role of Cryptography in Information Security 14](#_d4ler43belt)

[**III. Methodology 15**](#_x5joio95zzla)

[A. Conceptual Framework - IPO Chart (Input-Process-Output-Chart) 15](#_sg4r41xan6lg)

[B. Hierarchy Chart 16](#_qiclzmq5vr1m)

[C. Flowchart 18](#_eujw7yee8p1d)

[D. Pseudocode 36](#_mrsqxzag8ull)

[**IV. Results 48**](#_gydacwa730x8)

[**V. Discussion of Results 52**](#_3txgyjp5d02v)

[**VI. Analysis, Conclusion and Future Directives 53**](#_nudykep4mfzv)

[**References 54**](#_4b1mpr2mdh6l)

[**Appendices 56**](#_tkjjmffttl1n)

[A. User’s Manual 56](#_g578d84du6bz)

[B. Source Code 63](#_3myuguwqgmbg)

[C. Work Breakdown 73](#_plzz911wru5j)

[D. Personal Data Sheet 75](#_p14jpqox62xt)

**List of Figures**

***Figure 1:* Input-Process-Output Chart 11**

***Figure 2:* Hierarchy Chart 12**

***Figure 3:* Flowchart (main) 13**

***Figure 4:* Flowchart (shift) 14**

***Figure 5:* Flowchart (vigenere) 18**

***Figure 6:* Flowchart (rsa) 20**

***Figure 7:* Flowchart (quiz) 22**

***Figure 8:* Flowchart (shift\_enc) 23**

***Figure 9:* Flowchart (shift\_dec) 24**

***Figure 10:* Flowchart (vigenere\_key) 25**

***Figure 11:* Flowchart (vigenere\_enc) 26**

***Figure 12:* Flowchart (vigenere\_dec) 27**

***Figure 13:* Flowchart (prime\_check) 28**

***Figure 14:* Flowchart (prime\_key) 29**

***Figure 15:* Flowchart (rsa\_enc) 30**

***Figure 16:* Flowchart (rsa\_dec) 31**

***Figure 17:* Test Run 1 (Shift Cipher) 44**

***Figure 18:* Test Run 2 (Vigenere Cipher) 46**

***Figure 19:* Test Run 3 (RSA Cryptosystem) 47**

***Figure 20:* Test Run 4 (Quiz) 48**

# 

# **Introduction**

## Background of the Study

In 600-400 BC, ancient Greece was in constant war. And in war between equally strong men wielding equally deadly weapons, the key to turning the tides of battle in the favor of one side is intelligence. Soldiers need to be able to pass on information in the frontlines to their Generals so that they can counteract and secure victory. This was well before the invention of radios and the internet and as such, soldiers had to relay information in person, through word of mouth or writing. However, this method of relaying information is susceptible to interception, and intelligence falling into the wrong hands could prove detrimental to those fighting the war. Which is why the ancient Spartans came up with the Scytale. The scytale is a rod that is used in encrypting or decrypting secret messages. It is a rod that the soldiers can transcribe their message in and change the arrangement of the letters in the message. It is the first ever recorded use of cryptography in history.

Over the millennia, cryptography has undergone many changes and developments. From a simple rod to complex and intricate algorithms that allow for secure communication through insecure channels, they have developed to the point that they have become an essential part of today’s society, particularly, in the field of technology and cybersecurity. Cryptography has become involved in the emails that are sent, calls that are made, online banking transactions made, passwords that are kept, and more. However, in a survey conducted by Olmstead (2017), it was discovered that a majority of internet users are still unaware about certain cybersecurity issues and concepts. In an age where things are moving to a primarily online environment and where cybersecurity threats grow more advanced (Cisco Umbrella, 2023), it becomes all the more important that people are armed with an understanding and appreciation of the technology that is behind the security protocols that keep our online information safe and secure.

It is through these issues and concepts that the authors had the idea to develop *CRYPYTHONOLOGY: An Integration of Python Programming in Message Encryption and Decryption*, a python-powered platform where people are able to apply various cryptosystems to messages and gain a deeper understanding and appreciation of various cryptosystems.

## Problem Statement

This project mainly addresses the problem of the lack of basic knowledge, understanding, and application of cryptography among individuals. In this current situation where issues on cyber security are rampant, it is a necessity to cultivate the basic knowledge and comprehension of cryptography through this proposed program. By doing so, it will help people understand how to protect their personal information and digital assets from cyber threats.

## Objectives

**C.1 General Objective**

The general objective of this project deals with the extensive and comprehensive application of the various topics discussed in this course *LBYCPA1,* particularly the Python programming language, in establishing a diversified program on cryptography. Furthermore, this cryptography-based program aims to accurately simulate various cryptographic methods which can effectively educate the users in basic cryptography knowledge.

**C.2 Specific Objectives**

Under the project’s general objective, the project is specified into the following technical objectives:

1. To develop effective and efficient algorithms for each cryptographic method
2. To ensure the functionality of the program by correctly displaying the encrypted and decrypted message
3. To optimize the program’s performance by implementing appropriate and efficient data structures
4. To establish an operative score tracking system for the program’s ‘Quiz’ component
5. To implement and utilize built-in Python modules in the overall program development

## 

## Significance of the Project

The features of this project intends to establish a profound impact in society through equipping individuals with the knowledge and skills necessary in various situations revolving around cyber threats. These include securing personal information, business-related information, financial information, and other kinds of sensitive and valuable information. With this, this project aims to establish a knowledgeable and adaptive society through learning the fundamentals of cryptography, its applications, and its importance in today's digital age. As a result, it could initiate developments in reducing cyberattacks for individuals, businesses, and organizations which contribute to the growth, safety, and progress of the society.

# **Review of Related Literature**

## Conceptualization of Cryptography and Cryptosystems

Even before the Scytale of the ancient Greeks, the act of concealing messages from others have long been commonplace. They concealed messages by wrapping them in javelins, stuffing them in live rabbits, tying them to pigeons, using invisible inks, and more. But it was not until the invention of the Scytale that a standardized method of obscuring a message was invented and recorded in history.

### Shift Cipher

Shortly after the invention of the Scytale, Julius Caesar, a roman general, adopted the use of shift ciphers extensively, specifically with a shift of three, to secure messages of military significance. So much so that shift ciphers with that key came to be known as Caesar ciphers.

A shift cipher, also known as a Caesar cipher, a type of substitution cipher, is one of the simplest and most widely adopted cipher systems. It involves replacing each letter in the plaintext *K* places down the alphabet. For example, using a shift key of 2 will shift the letter C to the first letter, followed by all of the other letters, with the letter A being shifted to X and B to Z.

*CDEFGHIJKLMNOPQRSTUVWXYZAB*

*ABCDEFGHIJKLMNOPQRSTUVWXYZ*

Using a shift cipher with this key, the plaintext *Python* can be encrypted to *Ravjqp*.

This cipher system would have proved itself reasonably secure during the time of Julius Caesar as his enemies would not have been literate, let alone be able to consider cryptography. Nowadays, however, the cipher can be easily broken. Everyone is literate and shift ciphers can be broken at a glance. One can look at the frequency that the letters occur and deduce the *K* value that the plaintext was shifted with. With only a limited number of letters in an alphabet, one could also decode the cipher in what is known as a “brute force attack” where a snippet of the ciphertext is tested for all values of *K* until the plaintext is found.

### Vigenère Cipher

With how easily decodable Shift ciphers can be, a more secure method of cryptography ought to have been conceptualized. In shift ciphers, substitution is monoalphabetic, meaning that each letter in the plaintext is substituted for a fixed letter in the ciphertext key. An element found in Vigenère ciphers not found in shift ciphers is that substitutions are polyalphabetic, meaning that it is possible for different letters in the ciphertext to refer to the same letter in the plaintext. In the 16th century, Italian cryptographers Giovan Bellaso and Blaise de Vigenère invented such polyalphabetic ciphers. The Vigenère cipher is a type of substitution cipher where the plaintext is obscured using a keyword or phrase that is repeated for as many times as required to encipher the plaintext.

| Plaintext | M | O | N | T | Y | P | Y | T | H | O | N |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *x* | 12 | 14 | 13 | 19 | 24 | 15 | 24 | 19 | 7 | 14 | 13 |
| Keyword | C | O | D | E | C | O | D | E | C | O | D |
| y | 2 | 14 | 3 | 4 | 2 | 14 | 3 | 4 | 2 | 14 | 3 |

In this exemple, the plaintext *Monty Python* is to be encoded using the keyword *Code*. Each letter in the alphabet is assigned a value where A is 0, B is 1, and so on. The same is done for the keyword and the resulting values are added and converted back into its corresponding value in the alphabet.

| *x* | 12 | 14 | 13 | 19 | 24 | 15 | 24 | 19 | 7 | 14 | 13 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| y | 2 | 14 | 3 | 4 | 2 | 14 | 3 | 4 | 2 | 14 | 3 |
| z | 14 | 2 | 16 | 23 | 0 | 3 | 1 | 23 | 9 | 2 | 16 |
| Ciphertext | O | C | Q | X | A | D | B | X | J | C | Q |

Using the Vigenère cipher, plaintext *Monty Python* can be enciphered to *Ocqxa Dbxjcq*.

To decrypt, the same procedure is followed except the keyword’s values are subtracted from the ciphertext to recover the plaintext. Cryptanalysis for the Vigenère cipher was significantly more difficult than for any other cryptosystem and for many years was considered to be indecipherable. The Vigenère cipher was the biggest leap in cryptography anyone has ever had for over 1,000 years. Nevertheless, in the 19th century, cryptanalysts found a weakness in the Vigenère cipher that allowed them to be cracked. A solution was found based on the fact that identical pairings of keyword and the plaintext generate the same ciphertext. The cryptanalyst can deduce the length of the key and attempt to solve each as simple substitution ciphers.

### RSA Cryptosystem

It was not until the 20th century that the next innovation in cryptography would be made. In the previously mentioned ciphering systems, the same key is used in both ciphering and deciphering the plaintext. The idea of using differing keys for ciphering and deciphering was introduced by mathematicians Hellman and Diffie from Stanford University. The idea is that a public key can be used to encipher a plaintext but only a secret, private key can be used to decipher it. Making it so that anyone who wishes to send sensitive information can encrypt that information using the public key and send it over insecure channels, and only the person holding the private key would be able to decrypt it. This method of encryption is also referred to as asymmetric cryptography. An example of this is the RSA Cryptosystem.

In order to acquire the public and private keys:

1. Take two large prime numbers, and , and compute .
2. Choose a number less than n and relatively prime to .
3. Find such that is divisible by .

for some integer k.

1. The public key is and the private key is .

Suppose we are to encrypt the plaintext *Python*.

1. The chosen value for and . Then,
2. Choose . This number is both less than 2537 and relatively prime to .
3. If chosen , .
4. Hence, the public key is (2537, 13) and the private key is (2357, 937).

To encrypt the message, translate *Python* into a sequence of integers where A is 1, and so on, and perform the following computation:

| P | 16 |  | 1841 |
| --- | --- | --- | --- |
| Y | 25 |  | 1433 |
| T | 20 |  | 0793 |
| H | 8 |  | 0156 |
| O | 15 |  | 2116 |
| N | 14 |  | 1616 |

Therefore, the message *Python* is equivalent to the cipher text 1841 1433 0793 0156 2116 1616.

The receiver of the encrypted message 1841 1433 0793 0156 2116 1616 will perform the following decryption:

| 1841 |  | 16 | P |
| --- | --- | --- | --- |
| 1433 |  | 25 | Y |
| 0793 |  | 20 | T |
| 0156 |  | 8 | H |
| 2116 |  | 15 | O |
| 1616 |  | 14 | N |

## 

## Role of Cryptography in Information Security

Cryptography is a critical component of information security as it provides techniques for secure communication over the internet. The RSA cryptosystem, which employs public and private key cryptography, is instrumental in securing data sent online. The concept of having a public and private key is used in various modern technologies requiring secure communication. It is through public and private key cryptography that people can send messages and emails, transact, browse, and share files safely and securely without worrying about attackers stealing or tampering with the data sent.

Public and private key cryptography are the foundation of Transport Layer Security (TLS) protocols. SSL/TLS protocols use public and private key encryption systems like the RSA cryptosystem to establish a secure connection between the client and the server. When a client connects to a server using SSL/TLS, the server sends its public key to the client, which is used to encrypt a randomly generated secret key. The encrypted secret key is then sent back to the server, which can use its private key to decrypt it and obtain the secret key. This secret key is then used to encrypt and decrypt all subsequent communication between the client and the server using symmetric encryption. In particular, a website secured by SSL/TLS protocols can be identified with a padlock icon that appears in the address bar of web browsers. Furthermore, you can safely and securely send sensitive information through a site when the site you are accessing has HTTPS as part of its URL.

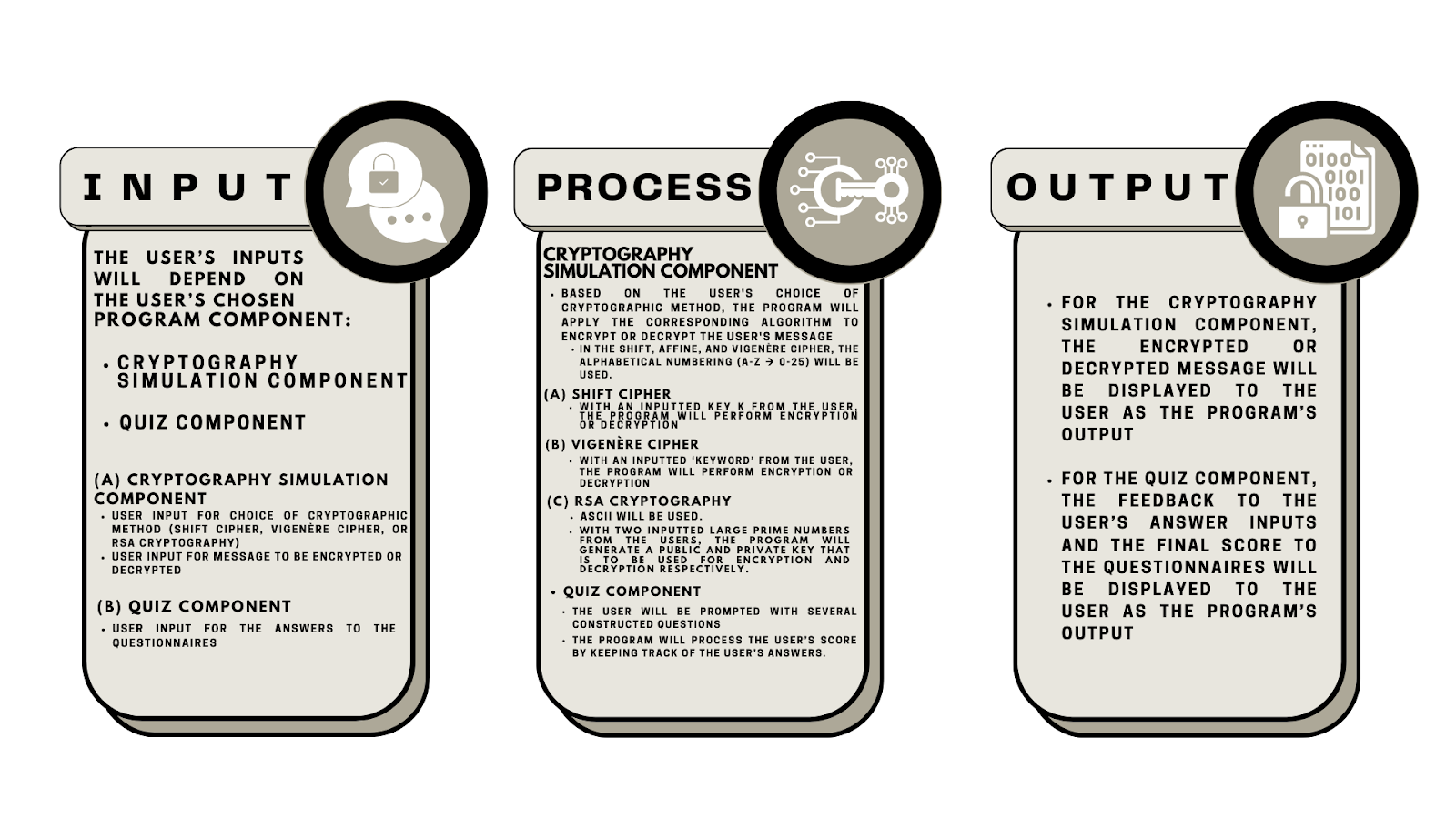
To ensure that the website is legitimate and has not been tampered with, SSL/TLS certificates are used. SSL/TLS certificates are digital documents that verify the identity of a website or web server and are issued by trusted third-party organizations called Certificate Authorities (CAs). These certificates, technologies, and other developments made in relation to cryptography are what allow users to verify that they are communicating with the intended website and not a fake website created by attackers to steal sensitive information.

# **Methodology**

In the overall development of the project, the methodology is essential to ensure that the project is carried out in a structured and systematic manner. The methodology includes a conceptual framework through the Input-Process-Output (IPO) Chart; hierarchy chart; flowchart; and pseudocode. Each of these components serves a unique purpose in the accomplishment of the project.

## Conceptual Framework - IPO Chart (Input-Process-Output-Chart)

***Figure 1:*** Input-Process-Output Chart



In the program’s development, three key components are typically identified: Input, Process, and Output.

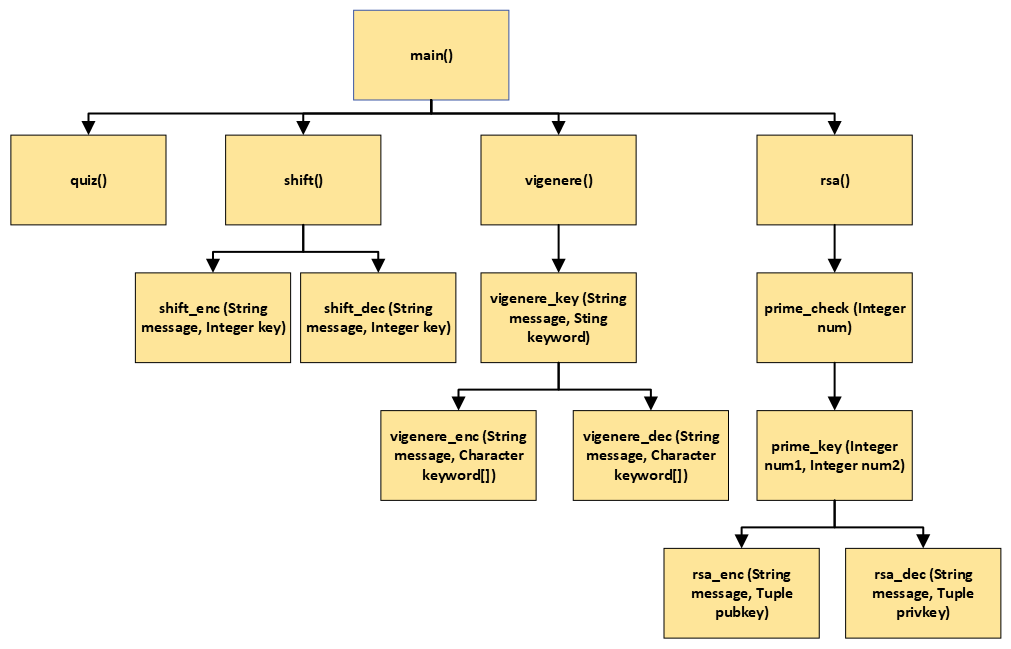
INPUT: The proposed program consists of a variety of possible inputs depending on the user’s chosen component. In the ‘Cryptography Simulation Component’, where the simulation of cryptographic methods are set, the user’s inputs include the user’s preferred cryptographic method and the message that the user wants to process using that particular cryptographic method. As for the ‘Quiz Component’, where the questionnaires are situated, the user’s inputs include the answers of the user to each question.

PROCESS: In each component of the program, complex processes and algorithms are performed. First, in the ‘Cryptography Simulation Component’, the processes involve the encryption and decryption algorithm of each cryptographic method. Furthermore, this process requires additional values and inputs from the user in order to successfully simulate the encryption and decryption processes. For the shift cipher, the algorithm requires a *key* value *K* in order to perform the cryptographic method. For vigenère cipher, the algorithm requires a *keyword* that will serve as the basis for encryption and decryption. And for RSA cryptosystem, the algorithm requires two large prime numbers as inputs. From these inputs, the algorithm will proceed in generating the public and private keys for encryption and decryption respectively. Additionally, the mentioned cryptographic methods will incorporate the alphabetical numbering (A-Z , 0-25) and ASCII for character conversion purposes. On the other hand, the processes within the ‘Quiz Component’ include the continuous gathering and prompting of questions to the user. Furthermore, the embedded processes shall attentively monitor the user’s score in the questionnaire.

OUTPUT: Since the program has two components, two different outputs must also be generated. For the ‘Cryptography Simulation Component’, the encrypted or decrypted message serves as the program's output. On the other hand, the output for the ‘Quiz Component’ consists of the feedback to each of the user’s answers and the obtained final score from the answered questionnaire. The feedback contains the evaluation of the user’s answer and an explanation of how the correct answer was obtained.

## Hierarchy Chart

***Figure 2:*** Hierarchy Chart

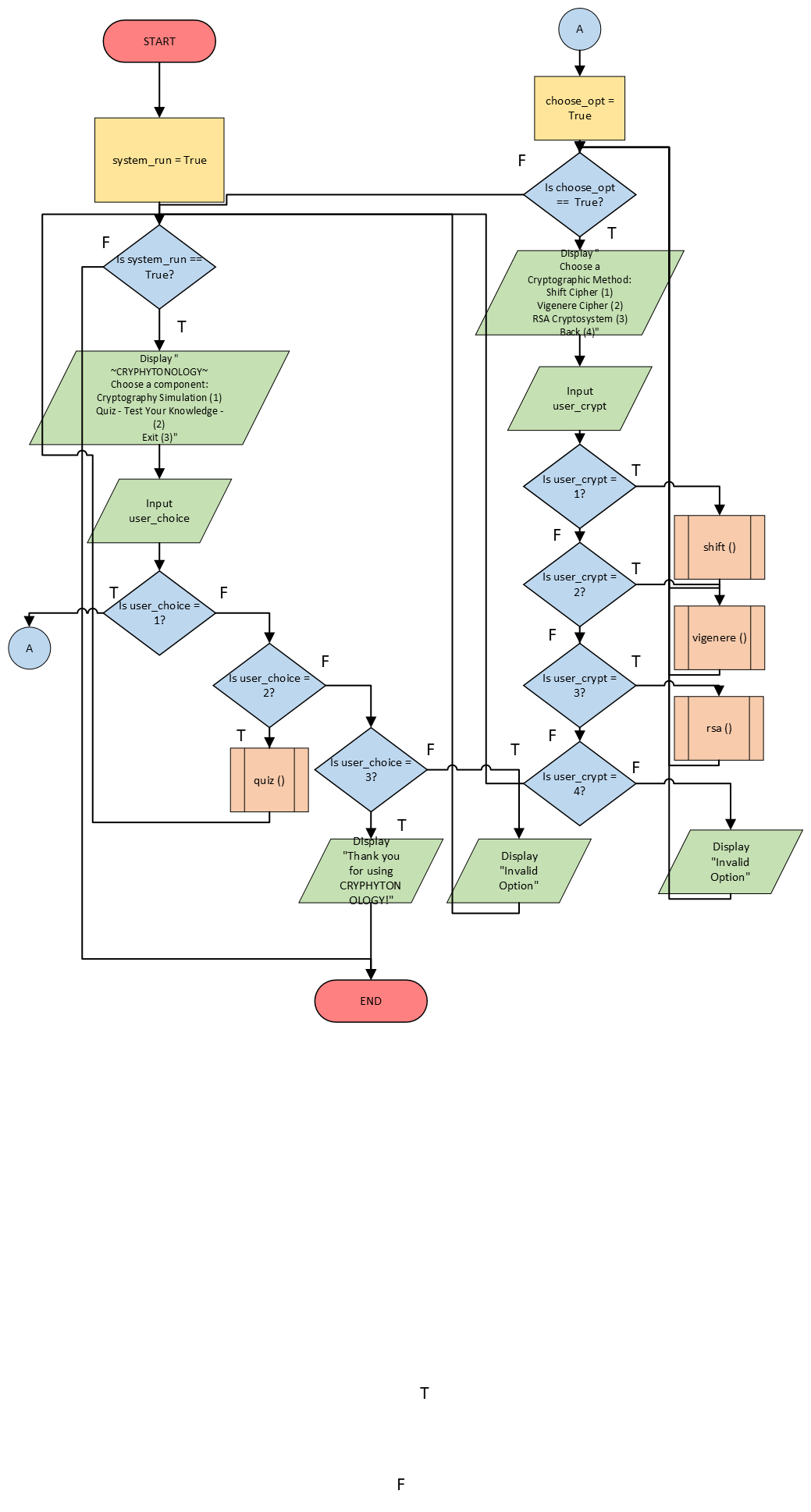
****

In the hierarchy chart, it can be observed that the *main* function is branched into four components namely *shift, vigenere, rsa,* and *quiz*. This is related to the program’s main menu in which the user is able to choose between the three cryptographic methods and the quiz component. The *shift* component is branched into *shift\_enc* and *shift\_dec*, which has a string and integer parameter. It is branched into two modules since cryptography involves encryption and decryption. Under the *vigenere* component, it is first branched into *vigenere\_key*, which uses two string parameters, and then it is branched into *vigenere\_enc* and *vigenere\_dec*, which has a string and character array parameter. Similar to the *shift* component, it was branched into two modules due to its encryption and decryption processes. Yet before these processes, it will first pass through the *vigenere\_key* module in order to generate the key to be utilized in the succeeding modules. For the *rsa* component, it first branches to *prime\_check* which takes one integer, then proceeds to *prime\_key* which takes two integers. These two modules focus on generating keys for encryption and decryption if the two integers in *prime\_key* are prime numbers which are analyzed by the *prime\_check* module. After this, *prime\_key* will be branched into *rsa\_enc* and *rsa\_dec* containing a string and tuple parameter.

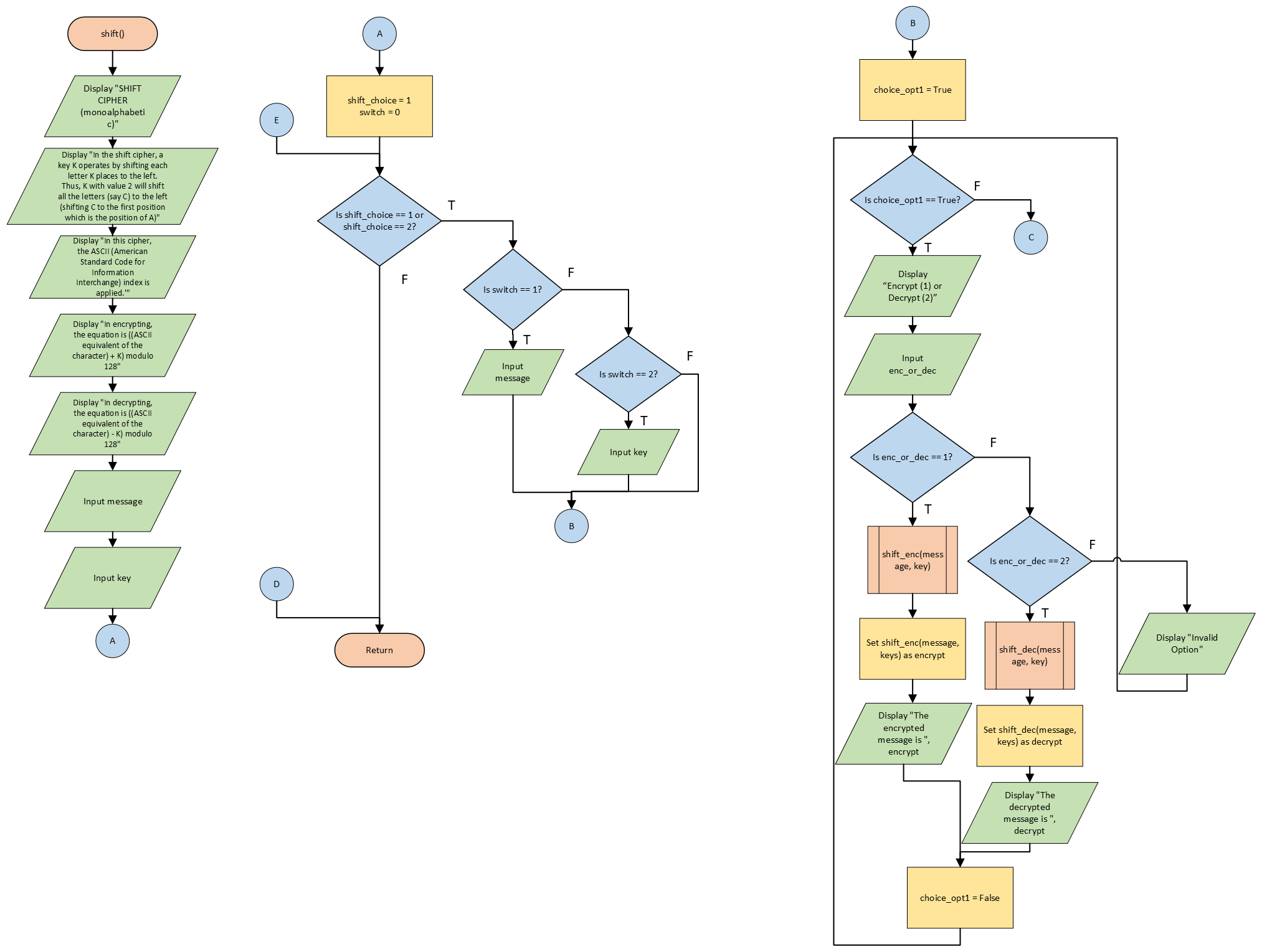
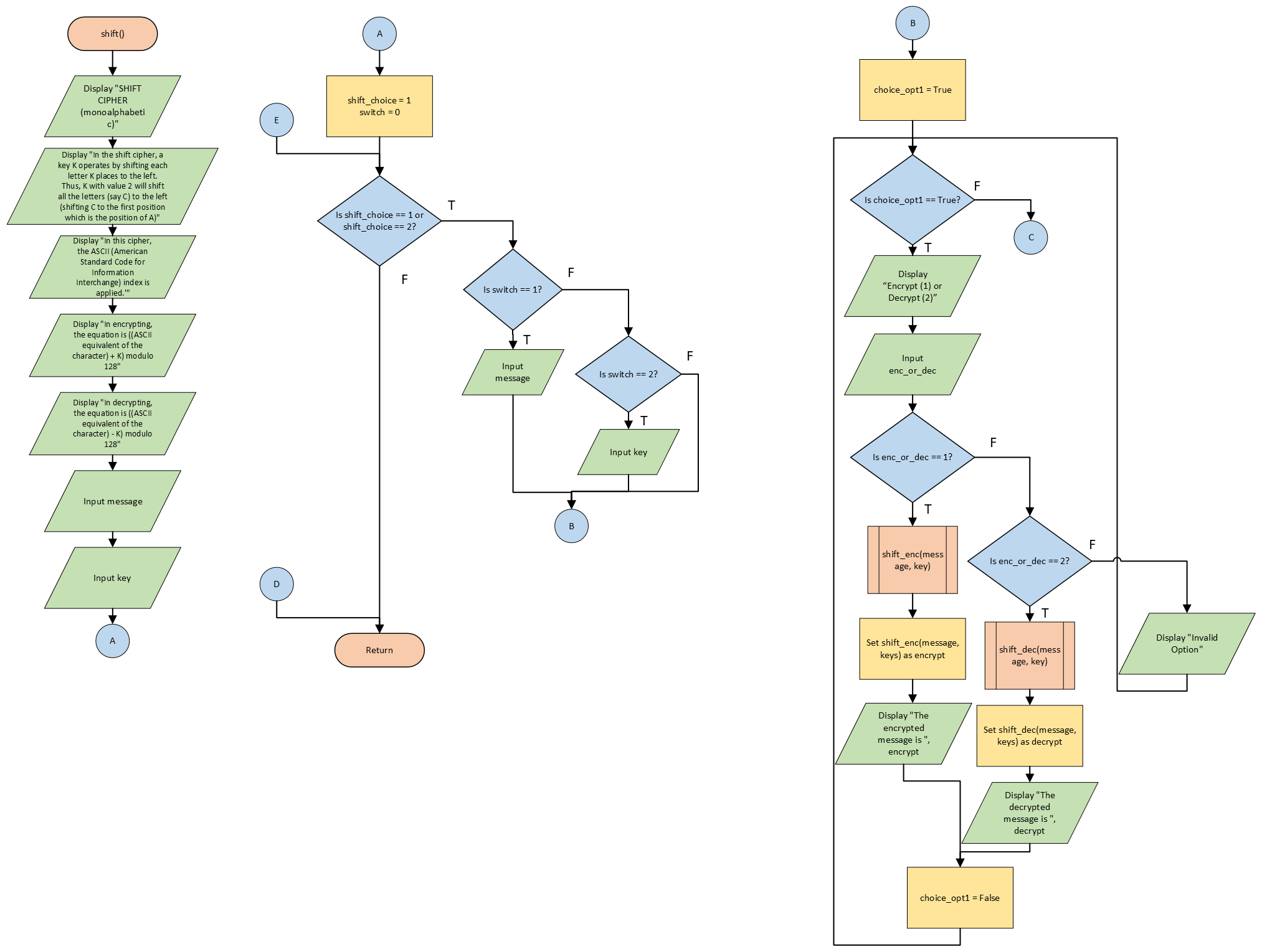
## Flowchart

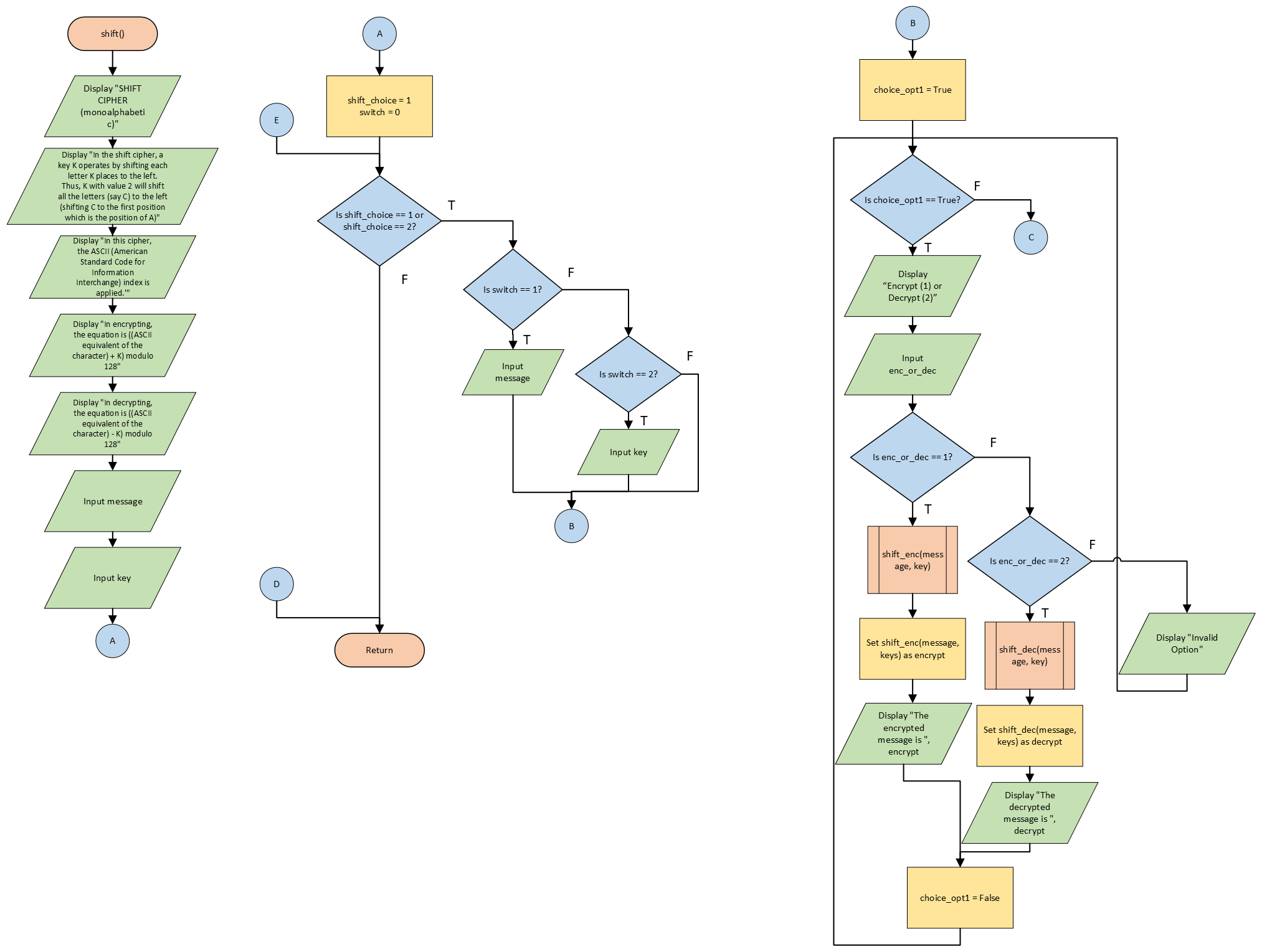
In the process of developing programs, graphical representations of the program’s algorithm, also known as flowcharts, are necessary as it guides the researchers through outlining the logic and flow behind the algorithm. In the context of this project, the researchers have graphically represented the program’s algorithm through 14 flowcharts, one for the main module, and the other 13 for the numerous subroutines.

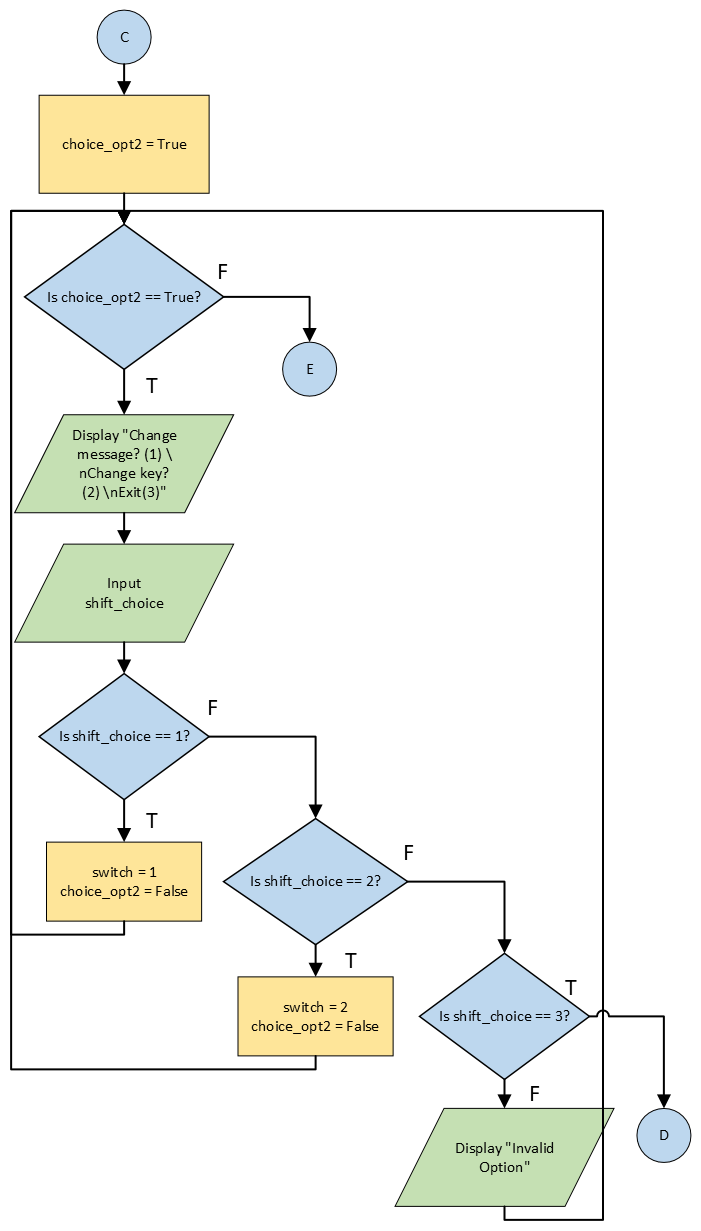
***Figure 3:*** Flowchart (main)



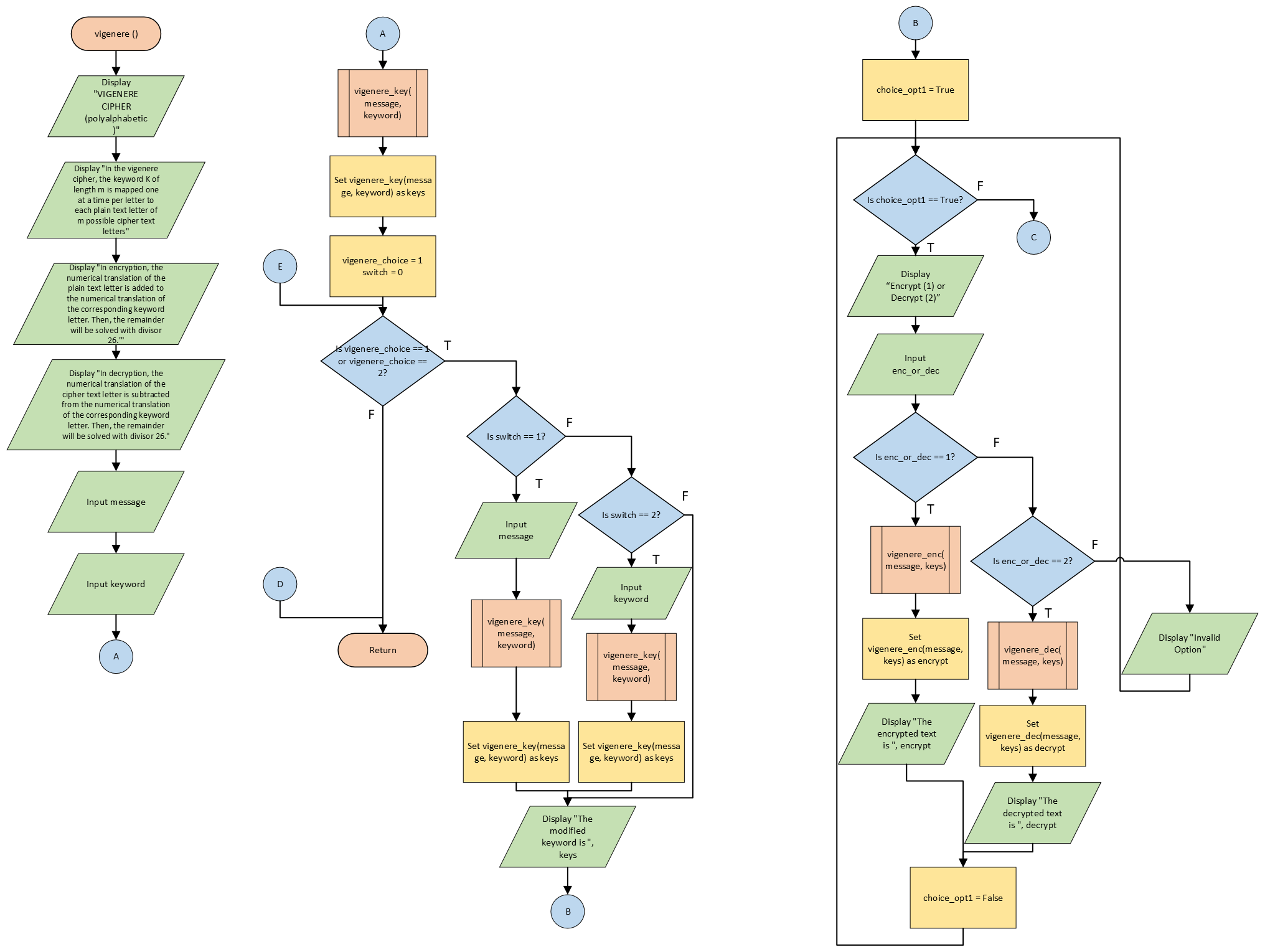
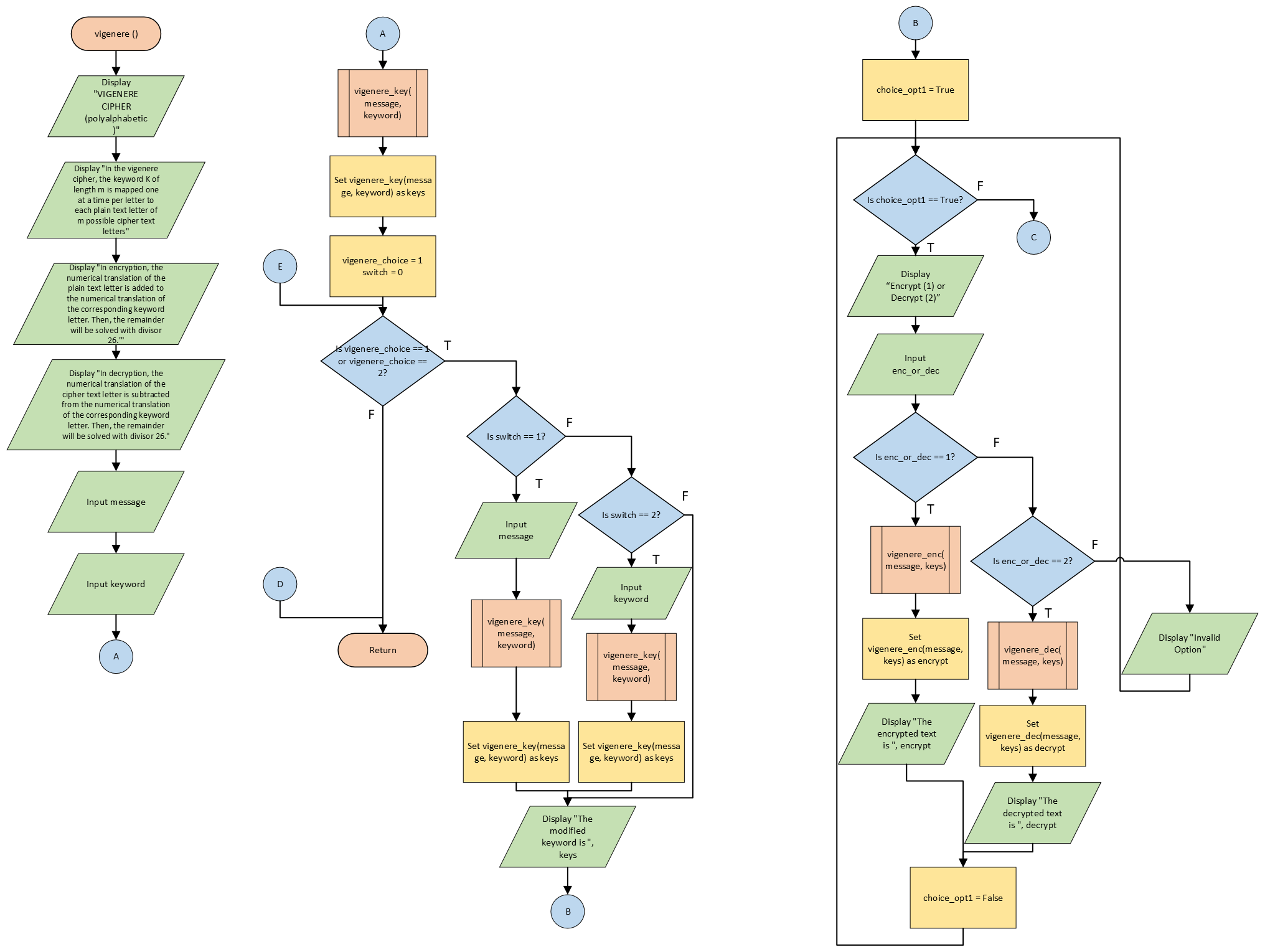
***Figure 4:*** Flowchart (shift)

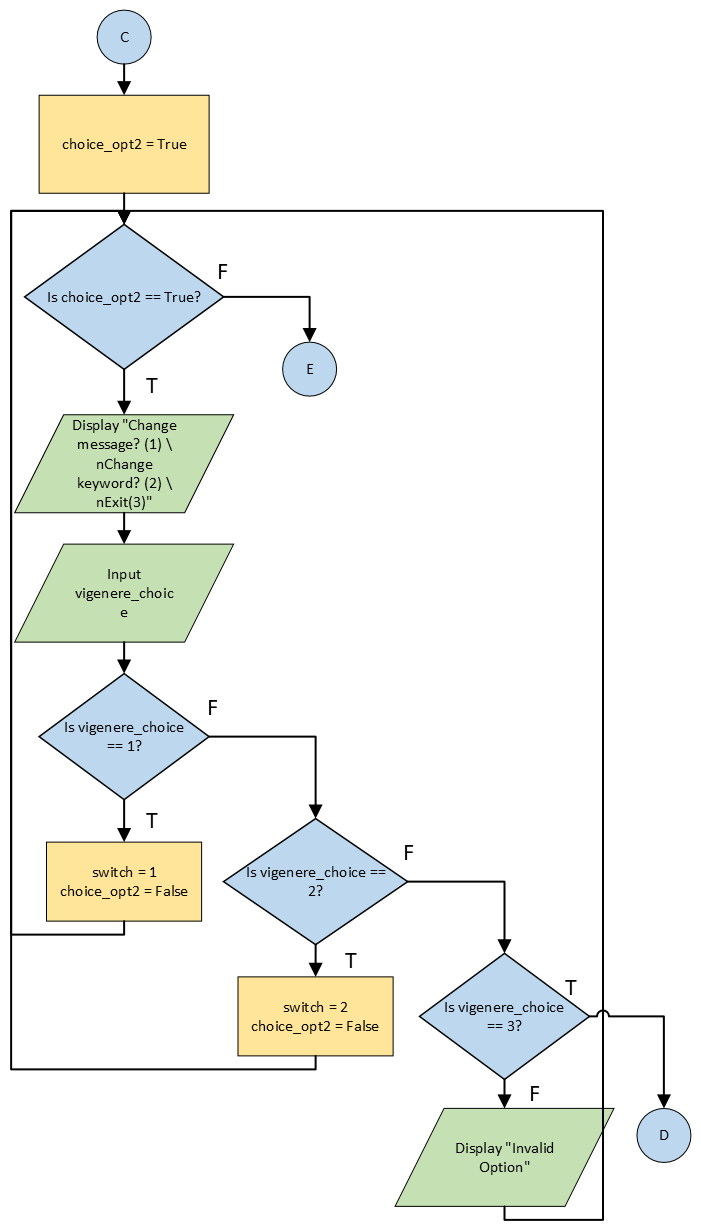
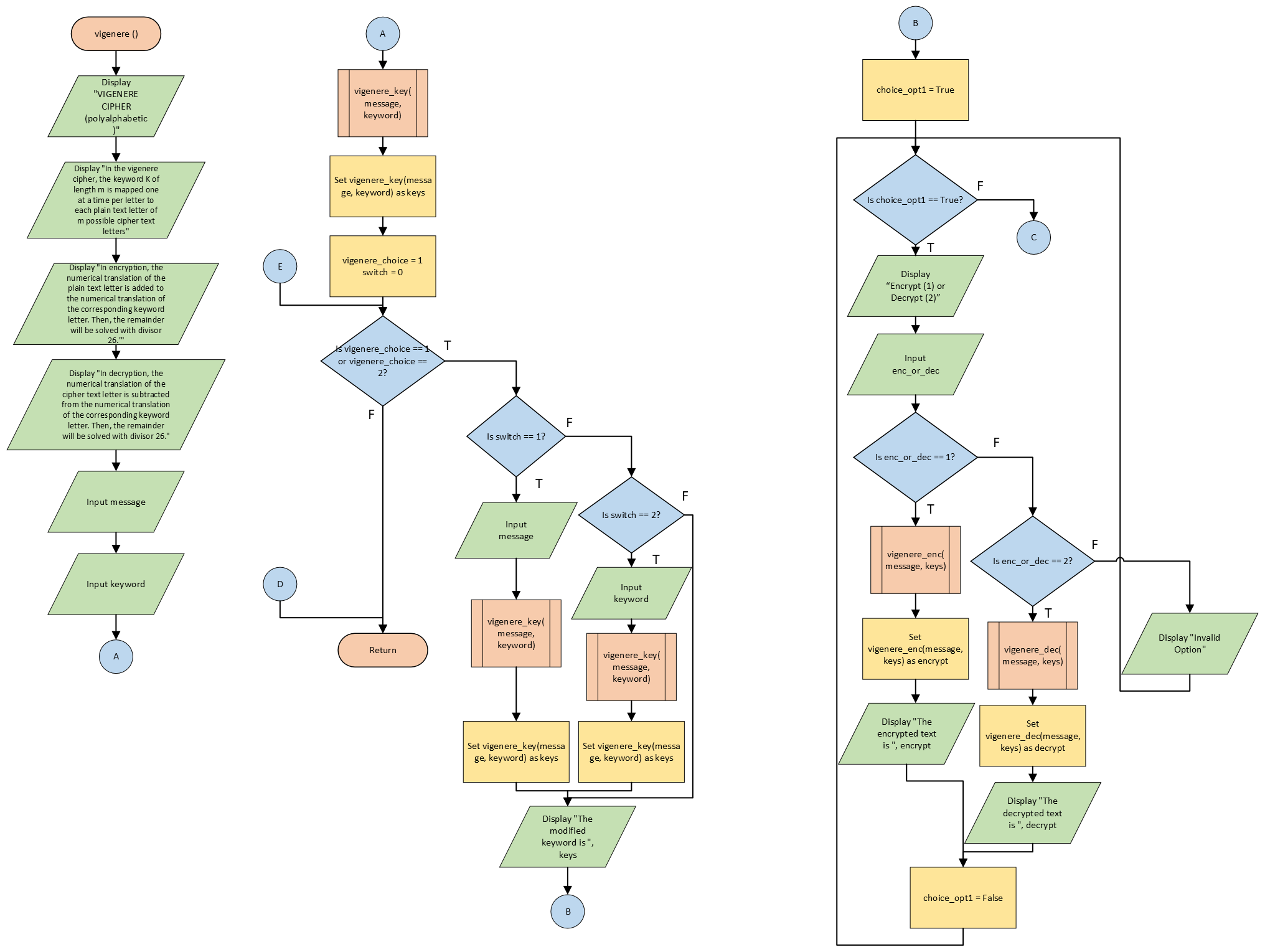




******

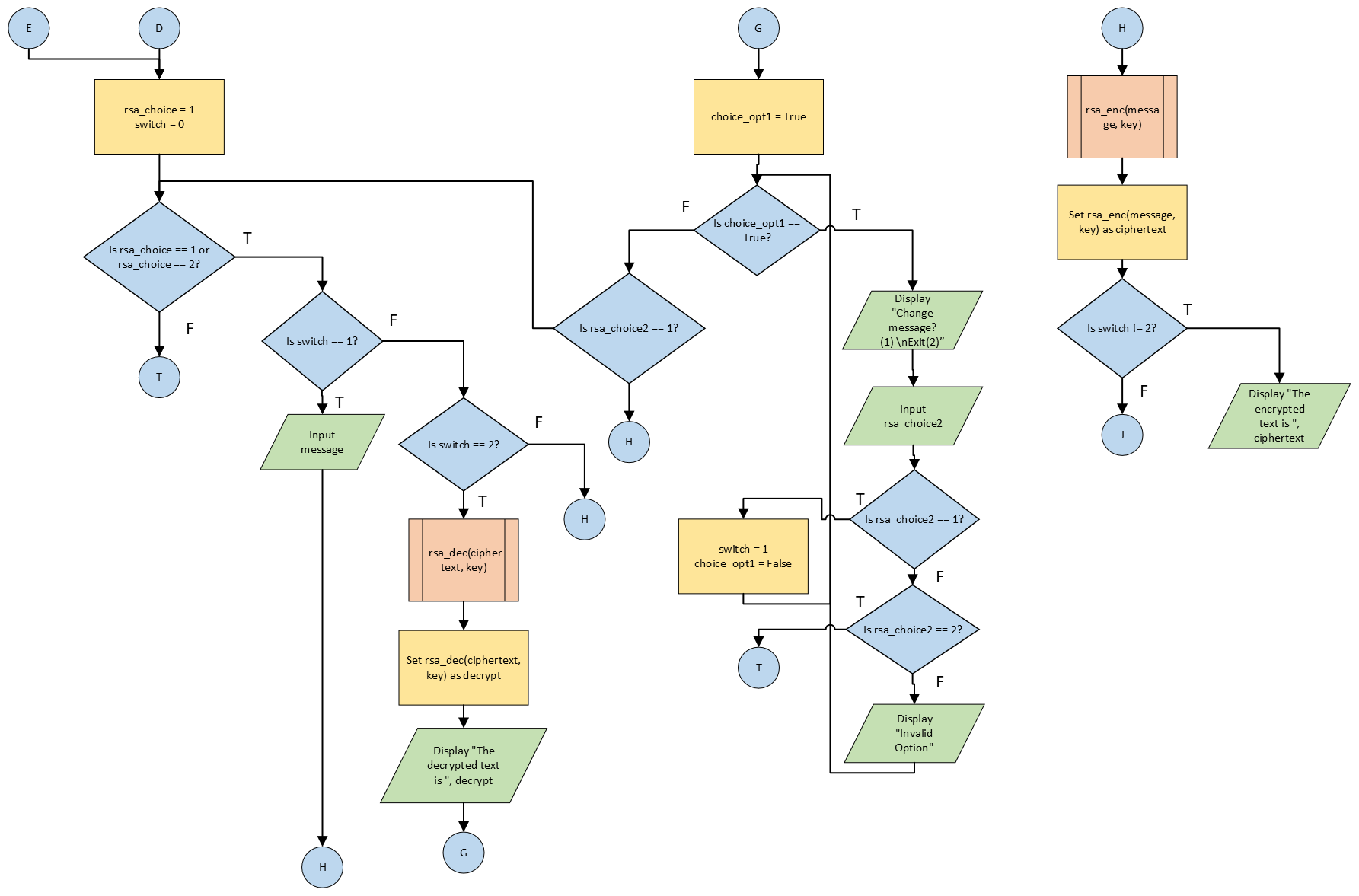
***Figure 5:*** Flowchart (vigenere)

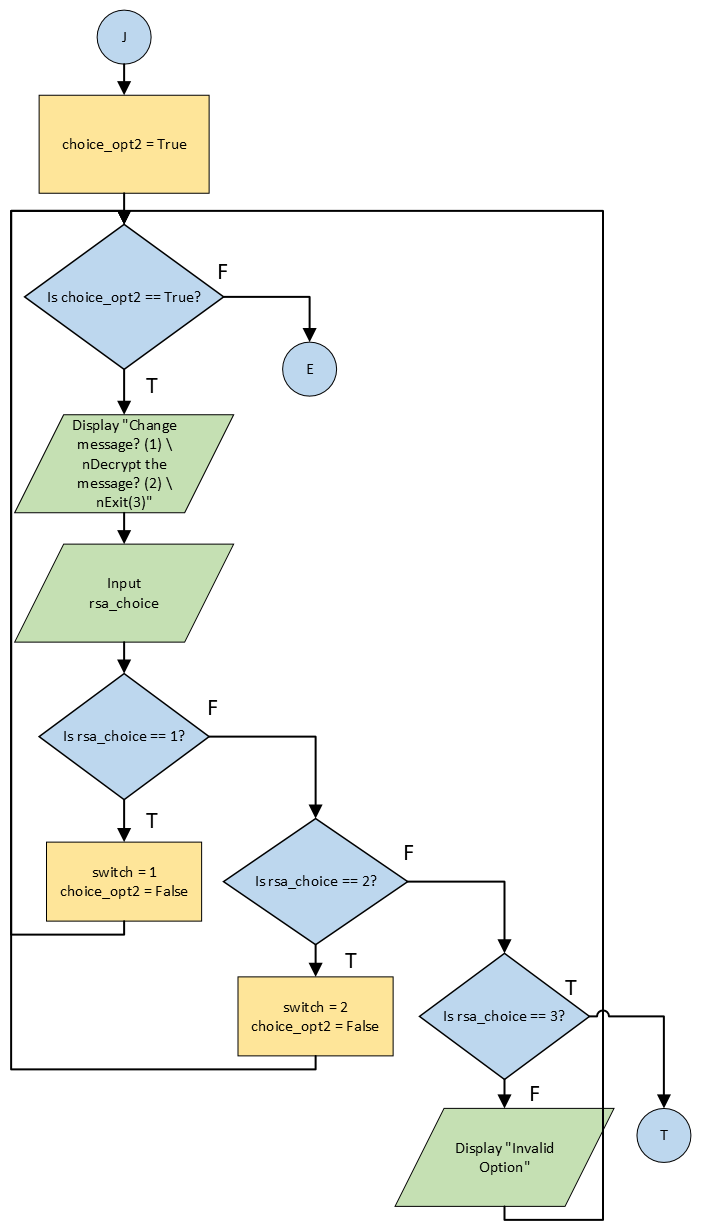




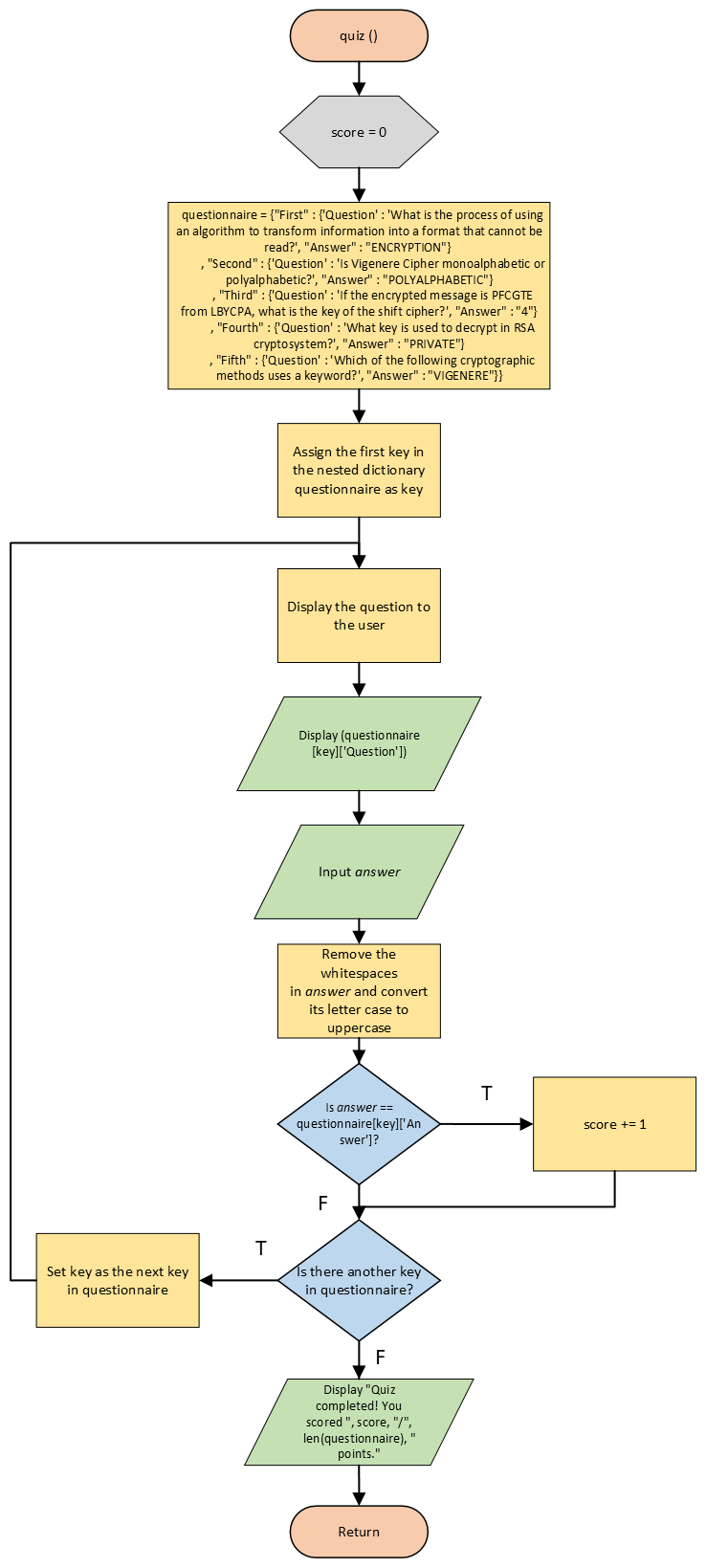
***Figure 6:*** Flowchart (rsa)



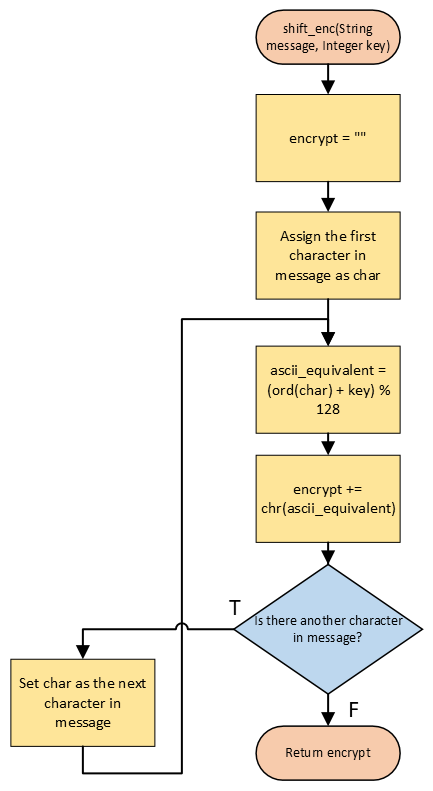


******

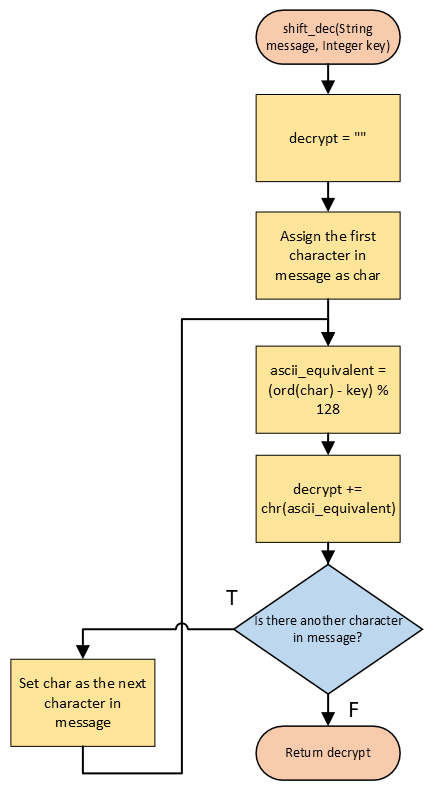
***Figure 7:*** Flowchart (quiz)



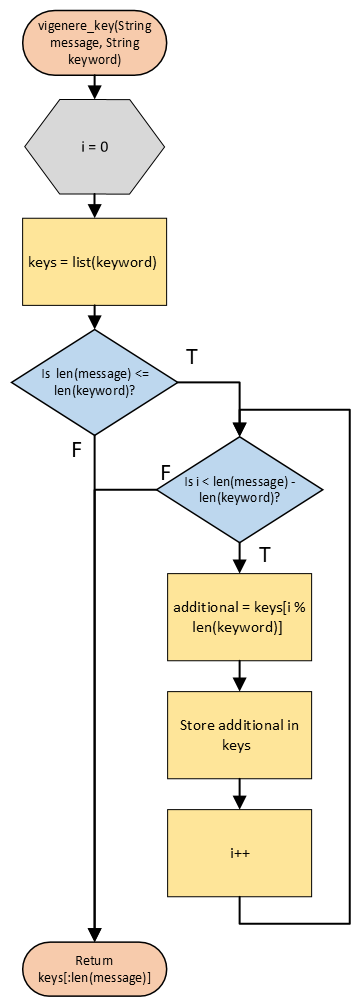
***Figure 8:*** Flowchart (shift\_enc)

******

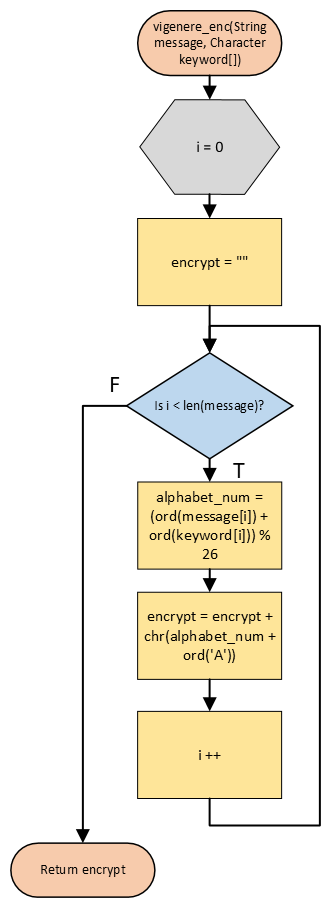
***Figure 9:*** Flowchart (shift\_dec)



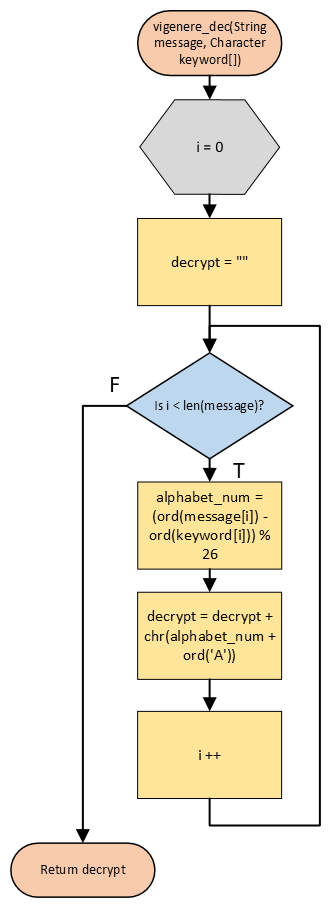
***Figure 10:*** Flowchart (vigenere\_key)



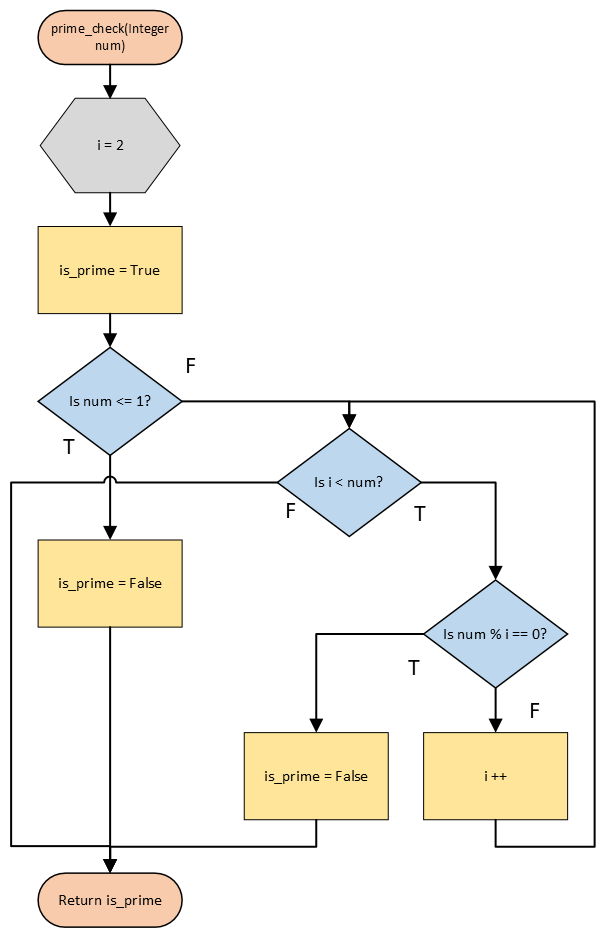
***Figure 11:*** Flowchart (vigenere\_enc)



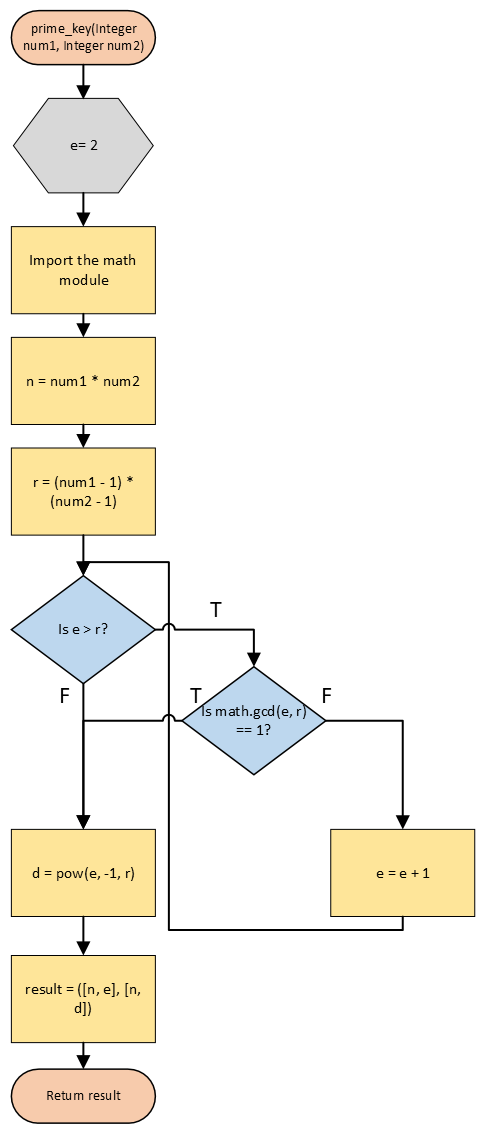
***Figure 12:*** Flowchart (vigenere\_dec)



***Figure 13:*** Flowchart (prime\_check)



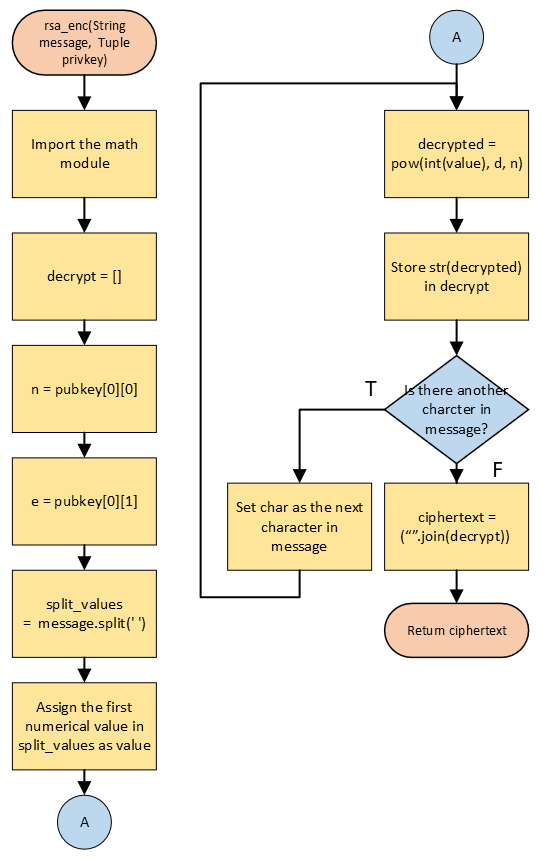
***Figure 14:*** Flowchart (prime\_key)



***Figure 15:*** Flowchart (rsa\_enc)



***Figure 16:*** Flowchart (rsa\_dec)



## Pseudocode

The pseudocode is an essential part of the methodology as it provides a structured and high-level representation of the proposed algorithm. In the project’s development, 14 modules, including the main module, are incorporated. These modules have their respective algorithms which are represented in the form of a pseudocode.

**Module main()**

1. *// This module is a menu-based system that allows users to simulate the different given cryptographic methods and take a quiz to test their knowledge. It continuously loops until the user chooses to exit the program. Users can select options such as cryptography simulation, quiz or exit. Depending on their choice, certain statements will be performed. Choosing cryptography simulation will direct the user to another set of options containing the three cryptographic methods. Choosing quiz will direct the user to the quiz component which contains a set of questions in which the user needs to answer. Choosing exit will terminate the system. In the module, invalid options will force the user to input another value.*
2. Module main()
3. Declare Integer user\_choice, user\_crypt
4. Declare Boolean system\_run, choose\_opt
5. Set system\_run = True
6. While system\_run
7. Display “~CRYPHYTONOLOGY~\n\nChoose a component: \nCryptography Simulation (1) \nQuiz - Test Your Knowledge (2) \nExit (3)”
8. Display “Input number: “
9. Input user\_choice
10. If user\_choice == 1
11. Set choose\_opt = True
12. While choose\_opt
13. Display Choose a Cryptographic Method: \nShift Cipher (1) \nVigenere Cipher (2) \nRSA Cryptosystem (3) \nBack (4)”
14. Display “Input number: “
15. Input user\_crypt
16. If user\_crypt == 1
17. Call shift()
18. Else if user\_crypt == 2
19. Call vigenere()
20. Else if user\_crypt == 3
21. Call rsa()
22. Else if user\_crypt == 4
23. Break Loop
24. Else
25. Display “Invalid Option\n”
26. End if
27. End while
28. Else if user\_choice == 2
29. Call quiz()
30. Else if user\_choice == 3
31. Display “Thank you for using CRYPHYTONOLOGY!”
32. Break Loop
33. Else
34. Display “Invalid Option”
35. End if
36. End while
37. End module

**Module shift()**

1. // *This module implements the interface for the shift cipher*
2. Module shift()
3. Declare Integer shift\_choice, switch, key, enc\_or\_dec
4. Declare Boolean choice\_opt1, choice\_opt2
5. Declare String message, encrypt, decrypt
6. Display “SHIFT CIPHER (monoalphabetic)\n”
7. Display “In the shift cipher, a key K operates by shifting each letter K places to the left. Thus, K with value 2 will shift all the letters (say C) to the left (shifting C to the first position which is the position of A)”
8. Display “\nIn this cipher, the ASCII (American Standard Code for Information Interchange) index is applied.”)
9. Display “\nIn encrypting, the equation is ((ASCII equivalent of the character) + K) modulo 128”
10. Display “'In decrypting, the equation is ((ASCII equivalent of the character) - K) modulo 128”
11. Display “Input message: “
12. Input message
13. Display “Input key: “
14. Input key
15. Set shift\_choice = 1
16. Set switch = 0
17. While shift\_choice == 1 or shift\_choice == 2
18. If switch == 1
19. Display “Input message: “
20. Input message
21. End if
22. If switch == 2
23. Display “Input key: “
24. Input key
25. End if
26. Set choice\_opt1 = True
27. While choice\_opt1
28. Display “Encrypt (1) or Decrypt (2)”
29. Display “Input number: “
30. Input enc\_or\_dec
31. If enc\_or\_dec == 1
32. encrypt = shift\_enc(message, key)
33. Display “The encrypted message is “, encrypt
34. Set choice\_opt1 = False
35. Else if enc\_or\_dec == 2
36. decrypt = shift\_dec(message, key)
37. Display “The decrypted message is “, decrypt
38. Set choice\_opt1 = False
39. Else
40. Display “Invalid Option\n”
41. End if
42. End while
43. Set choice\_opt2 = True
44. While choice\_opt2
45. Display “Change message? (1) \nChange key? (2) \nExit(3)”
46. Display “Input number: “
47. Input shift\_choice
48. If shift\_choice == 1
49. Set switch = 1
50. Set choice\_opt2 = False
51. Else if shift\_choice == 2
52. Set switch = 2
53. Set choice\_opt2 = False
54. Else if shift\_choice == 3
55. Return
56. Else
57. Display “Invalid Option\n”
58. End if
59. End while
60. End while
61. End module

**Module vigenere()**

1. // *This module implements the interface for the vigenere cipher*
2. Module vigenere()
3. Declare Integer vigenere\_choice, switch, enc\_or\_dec
4. Declare Boolean choice\_opt1, choice\_opt2
5. Declare String message, keyword, encrypt, decrypt
6. Declare Character keys[]
7. Display “VIGENERE CIPHER (polyalphabetic)\n”
8. Display “In the vigenere cipher, the keyword K of length m is mapped one at a time per letter to each plain text letter of m possible cipher text letters\n)”
9. Display “In this cipher, the index used is 'A-Z' -> '0-25'\n”)
10. Display “In encryption, the numerical translation of the plain text letter is added to the numerical translation of the corresponding keyword letter. Then, the remainder will be solved with divisor 26."”
11. Display “'In decryption, the numerical translation of the cipher text letter is subtracted from the numerical translation of the corresponding keyword letter. Then, the remainder will be solved with divisor 26.”
12. Display “Input message: “
13. Input message
14. Display “Input keyword: “
15. Input keyword
16. keys = vigenere\_key(message, keyword)
17. Set vigenere\_choice = 1
18. Set switch = 0
19. While vigenere\_choice == 1 or vigenere\_choice == 2
20. If switch == 1
21. Display “Input message: “
22. Input message
23. keys = vigenere\_key(message, keyword)
24. End if
25. If switch == 2
26. Display “Input keyword: “
27. Input keyword
28. keys = vigenere\_key(message, keyword)
29. End if
30. Display “The modified keyword is “, keys
31. Set choice\_opt1 = True
32. While choice\_opt1
33. Display “Encrypt (1) or Decrypt (2)”
34. Display “Input number: “
35. Input enc\_or\_dec
36. If enc\_or\_dec == 1
37. encrypt = vignere\_enc(message, keys)
38. Display “The encrypted text is “, encrypt
39. Set choice\_opt1 = False
40. Else if enc\_or\_dec == 2
41. decrypt = Call vigenere\_dec(message, keys)
42. Display “The decrypted text is “, decrypt
43. Set choice\_opt1 = False
44. Else
45. Display “Invalid Option\n”
46. End if
47. End while
48. Set choice\_opt2 = True
49. While choice\_opt2
50. Display “Change message? (1) \nChange keyword? (2) \nExit(3)”
51. Display “Input number: “
52. Input vigenere\_choice
53. If vigenere\_choice == 1
54. Set switch = 1
55. Set choice\_opt2 = False
56. Else if vigenere\_choice == 2
57. Set switch = 2
58. Set choice\_opt2 = False
59. Else if vigenere\_choice == 3
60. Return
61. Else
62. Display “Invalid Option\n”
63. End if
64. End while
65. End while
66. End module

**Module rsa()**

1. // *This module implements the interface for the rsa cryptosystem*
2. Module rsa()
3. Declare Integer rsa\_choice, switch, enc\_or\_dec, num1, num2, repeat\_choice
4. Declare Boolean choice\_opt1, choice\_opt2, user\_input, is\_prime1, is\_prime2, repeat
5. Declare String message, keyword, decrypt, ciphertext
6. Declare Tuple key
7. Display “RSA Cryptosystem (asymmetric)\n”
8. Display “In this cipher, the ASCII (American Standard Code for Information Interchange) index is applied.\n)”
9. Display “In generating the key, the following procedure is followed:”)
10. Display “1. Take two large prime numbers, p and q, and compute n = pq"
11. Display “'2. Choose a number e less than n and relatively prime to (p-1)(q-1)”
12. Display “3. Find another number d such that ed-1 is divisible by (p-1)(q-1)”
13. Display “4. The public key is (n, e) <- for encryption, and the private key is (n, d) <- for decryption\n”
14. Display “In encrypting, the equation C=M^e (mod n) is applied”
15. Display “In decrypting, the equation M = C^d (mod n) is applied”
16. Display “Input message: “
17. Input message
18. Set user input = True
19. While user\_input
20. Display “Input a large prime number: “
21. Input num1
22. Display “Input another large prime number: “
23. Input num2
24. is\_prime1 = prime\_check(num1)
25. is\_prime2 = prime\_check(num2)
26. repeat = False
27. If is\_prime1 == False and is\_prime2 == False
28. Display “Both inputs are not prime numbers\n”
29. repeat = True
30. Else if is\_prime1 == False and is\_prime2 == True
31. Display “First Input is not a prime number”
32. repeat = True
33. Else if is\_prime1 == True and is\_prime2 == False
34. Display “Second Input is not a prime number”
35. repeat = True
36. Else
37. key = prime\_key(num1, num2)
38. Display “Your public key is “, key[0]
39. Display “Your private key is “, key[1]
40. Break Loop
41. End if
42. If repeat
43. choice\_opt3 = True
44. While choice\_opt3
45. Display “Input again (1)? Or Exit (2)?”
46. Display “Input number: “
47. Input repeat\_choice
48. If repeat\_choice == 1
49. choice\_opt3 = False
50. Else If repeat\_choice == 2
51. Return
52. Else
53. Display “Invalid Option\n”
54. End if
55. End while
56. End if
57. End while
58. Set rsa\_choice = 1
59. Set switch = 0
60. While rsa\_choice == 1 or rsa\_choice == 2
61. If switch == 1
62. Display “Input message: “
63. Input message
64. End if
65. If switch == 2
66. decrypt = rsa\_dec(ciphertext, key)
67. Display “The decrypted text is “, decrypt
68. choice\_opt1 = True
69. While choice\_opt1
70. Display “Change message? (1) \nExit(2)”
71. Display “Input number: “
72. Input rsa\_choice2
73. If rsa\_choice2 == 1
74. switch = 1
75. choice\_opt1 = False
76. Else if rsa\_choice2 == 2
77. Return
78. Else
79. Display “Invalid Option\n”
80. End if
81. End while
82. If rsa\_choice2 == 1
83. Continue
84. End if
85. End if
86. ciphertext = rsa\_enc(message, key)
87. If switch != 2
88. Display “The encrypted text is “, ciphertext
89. End if
90. choice\_opt2 = True
91. While choice\_opt2
92. Display “Change message? (1) \nDecrypt the message? (2) \nExit(3)”
93. Display “Input number: “
94. Input rsa\_choice
95. If rsa\_choice == 1
96. Set switch = 1
97. Set choice\_opt2 = False
98. Else if rsa\_choice == 2
99. Set switch = 2
100. Set choice\_opt2 = False
101. Else if rsa\_choice == 3
102. Return
103. Else
104. Display “Invalid Option\n”
105. End if
106. End while
107. End while
108. End module

**Module quiz()**

1. // *This module presents a series of questions to the user and evaluates their answers based on the predefined set of correct answers. The total score obtained by the user will be displayed after completing the quiz*
2. Module quiz()
3. Declare Integer score = 0
4. Declare Dictionary questionnaire = {

"First" : {'Question' : 'What is the process of using an algorithm to transform information into a format that cannot be read?', "Answer" : "ENCRYPTION"} ,

"Second" : {'Question' : 'Is Vigenere Cipher monoalphabetic or polyalphabetic?', "Answer" : "POLYALPHABETIC"},

"Third" : {'Question' : 'If the encrypted message is PFCGTE from LBYCPA, what is the key of the shift cipher?', "Answer" : "4"},

"Fourth" : {'Question' : 'What key is used to decrypt in RSA cryptosystem?', "Answer" : "PRIVATE"} ,

"Fifth" : {'Question' : 'Which of the following cryptographic methods uses a keyword?', "Answer" : "VIGENERE"}

}

1. For Each key in questionnaire
2. Display questionnaire[key][‘Question’]
3. Input answer
4. Remove the whitespace in answer
5. Convert answer to uppercase
6. If answer == questionnaire[key][‘Answer’]
7. score = score + 1
8. Display “Correct!”
9. Else
10. Display “Incorrect!”
11. End if
12. End for
13. Display “Quiz completed! You scored “, score, “/”, len(questionnaire), “ points.”
14. End module

**Module shift\_enc(String message, Integer key)**

1. // *This module implements the encryption algorithm of the shift cipher*
2. Module shift\_enc(String message, Integer key)
3. Declare String encrypt = ""
4. Declare Character char
5. Declare Integer ascii\_equivalent
6. For Each char in message
7. ascii\_equivalent = (ord(char) + key) % 128
8. encrypt = encrypt + chr(ascii\_equivalent)
9. End for
10. Return encrypt
11. End module

**Module shift\_dec(String message, Integer key)**

1. // *This module implements the decryption algorithm of the shift cipher*
2. Module shift\_dec(String message, Integer key)
3. Declare String decrypt = ""
4. Declare Character char
5. Declare Integer ascii\_equivalent
6. For Each char in message
7. ascii\_equivalent = (ord(char) - key) % 128
8. decrypt = decrypt + chr(ascii\_equivalent)
9. End for
10. Return decrypt
11. End module

**Module vigenere\_key(String message, String keyword)**

1. // *This module generates the keyword to be used based from the message in performing the algorithm of the vigenere cipher*
2. Module vigenere\_key(String message, String keyword)
3. Declare Character keys[], additional
4. Declare Integer i
5. Set keys = list(keyword)
6. If len(message) == len(keyword)
7. Return keys
8. Else
9. For i = 0 to len(message) - len(keyword)
10. additional = keys[i % len(keyword)]
11. Store additional in keys
12. End for
13. Return keys
14. End if
15. End module

**Module vigenere\_enc(String message, Character keyword[])**

1. // *This module implements the encryption algorithm of the vigenere cipher*
2. Module vigenere\_enc(String message, Character keyword[])
3. Declare String encrypt
4. Declare Integer i, alphabet\_num
5. For i = 0 to len(message)
6. alphabet\_num = (ord(message[i]) + ord(keyword[i])) % 26
7. encrypt = encrypt + chr(alphabet\_num + ord('A'))
8. End for
9. Return encrypt
10. End module

**Module vigenere\_dec(String message, Character keyword[])**

1. // *This module implements the decryption algorithm of the vigenere cipher*
2. Module vigenere\_dec(String message, Character keyword[])
3. Declare String decrypt
4. Declare Integer i, alphabet\_num
5. For i = 0 to len(message)
6. alphabet\_num = (ord(message[i]) - ord(keyword[i])) % 26
7. decrypt = decrypt + chr(alphabet\_num + ord('A'))
8. End for
9. Return decrypt
10. End module

**Module prime\_check(Integer num)**

1. // *This module checks if the given number is a prime number or not.*
2. Module prime\_check(Integer num)
3. Declare Integer i
4. Declare Boolean is\_prime = True
5. If num <= 1
6. Set is\_prime = False
7. Else
8. For i = 2 to num
9. If num % i == 0
10. Set is\_prime = False
11. Break Loop
12. End if
13. End for
14. End if
15. Return is\_prime
16. End module

**Module prime\_key(Integer num1, Integer num2)**

1. // *This module generates the public and private keys for the RSA cryptosystem using the two number inputs num1 and num2*
2. Module prime\_key(Integer num1, Integer num2)
3. Declare Integer n, r, e, d
4. Declare Tuple result
5. Import the math module
6. Set n = num1 \* num2
7. Set r = (num1 - 1) \* (num2 - 1)
8. Set e = 2
9. While e > r
10. If math.gcd(e, r) == 1
11. Break Loop
12. Else
13. e = e + 1
14. End If
15. End while
16. d = pow(e, -1, r)
17. result = ([n, e], [n, d])
18. Return result
19. End module

**Module rsa\_enc(String message, Tuple pubkey)**

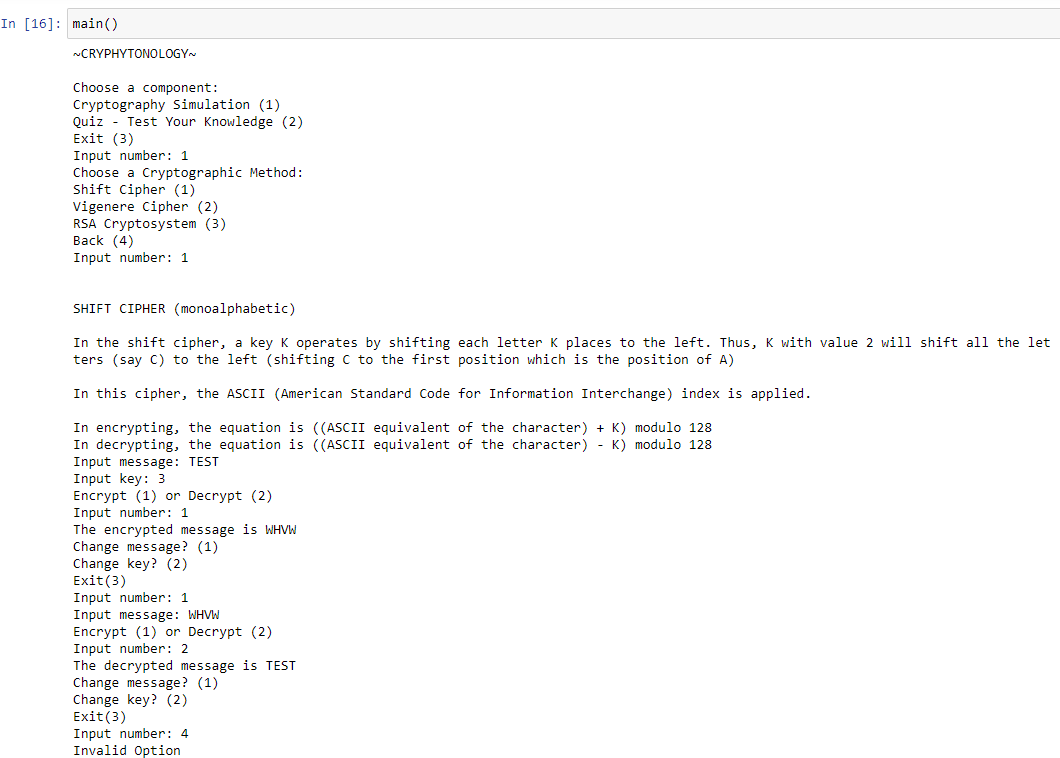
1. // *This module implements the encryption algorithm for the RSA cryptosystem.*
2. Module rsa\_enc(String message, Tuple pubkey)
3. Declare String encrypt[]
4. Declare Integer n, e, ascii\_equivalent, encrypted, ciphertext
5. Import the math module
6. Set n = pubkey[0][0]
7. Set e = pubkey[0][1]
8. For Each char in message
9. ascii\_equivalent = ord(char)
10. encrypted = pow(ascii\_equivalent, e, n)
11. Store encrypted in encrypt
12. End For
13. ciphertext = (“”.join(encrypt))
14. Return ciphertext
15. End module

**Module rsa\_dec(String message, Tuple privkey)**

1. // *This module implements the decryption algorithm for the RSA cryptosystem.*
2. Module rsa\_dec(String message, Tuple privkey)
3. Declare String decrypt[], split\_values[], value
4. Declare Integer n, d, decrypted, ciphertext
5. Set n = privkey[1][0]
6. Set d = privkey[1][1]
7. Split message with spaces and store the elements into split\_values
8. For Each value in split\_values
9. decrypted = pow(int(value), d, n)
10. Store decrypted in decrypt
11. End For
12. ciphertext = (‘’.join(decrypt))
13. Return ciphertext
14. End module

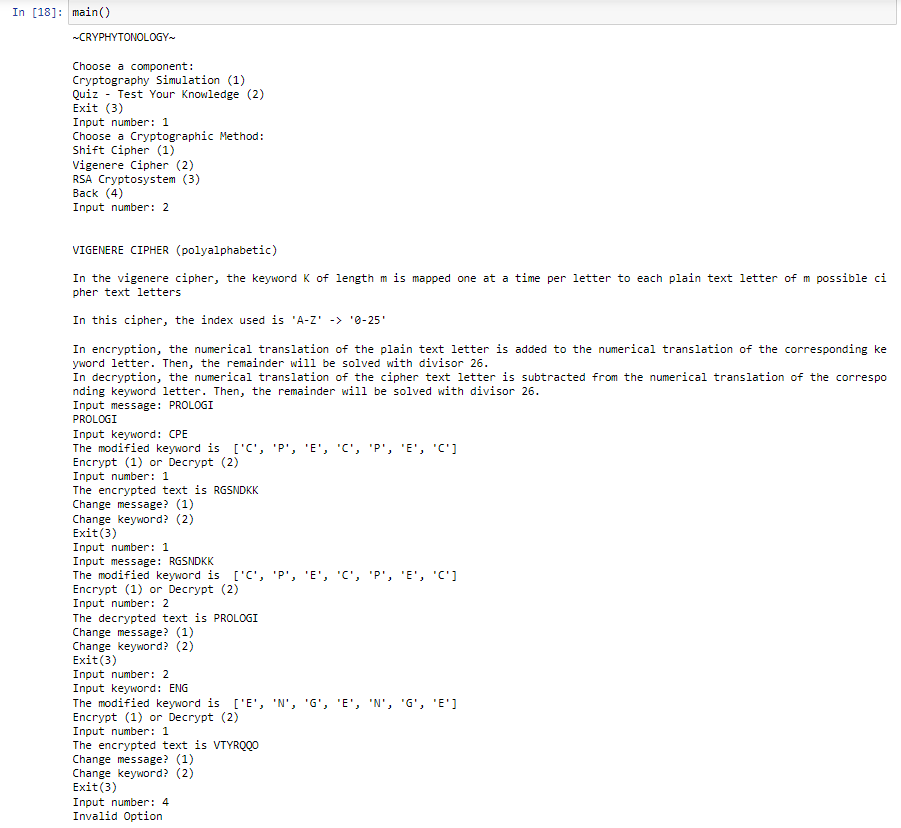
# **Results**

***Figure 16:*** Test Run 1 (Shift Cipher)



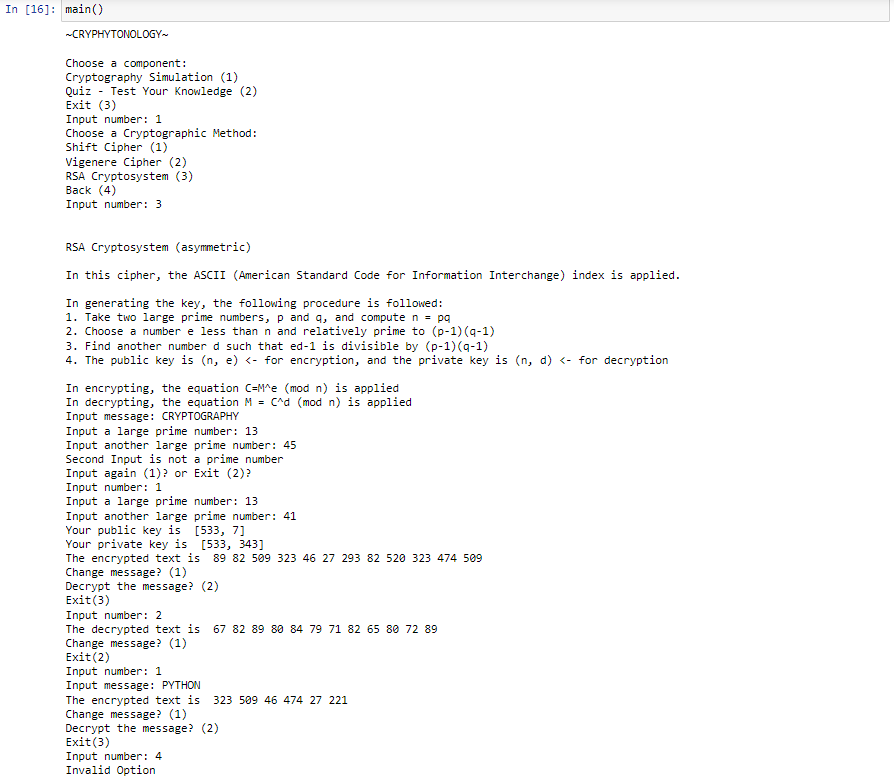


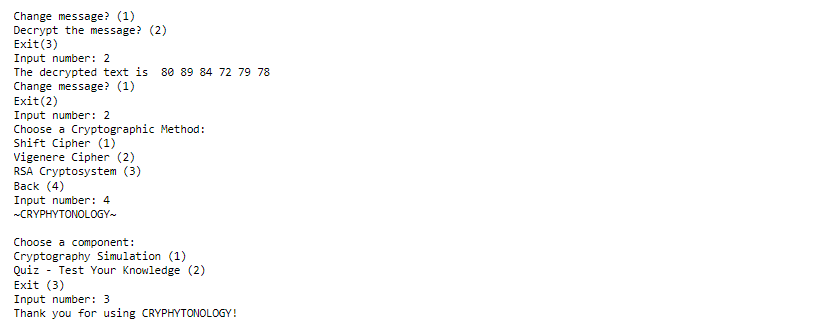
***Figure 17:*** Test Run 2 (Vigenere Cipher)



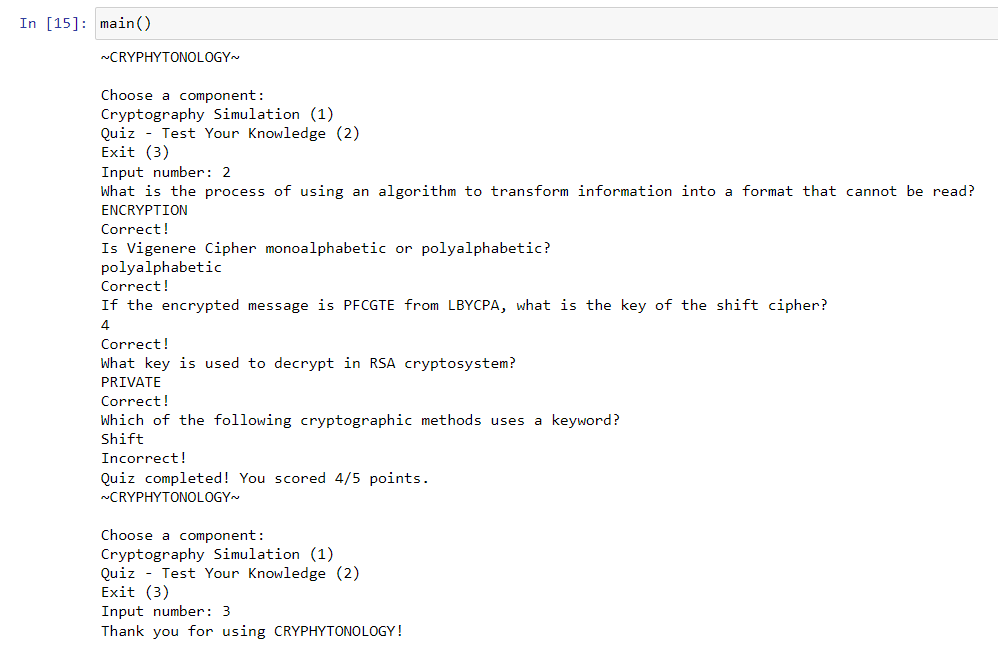


***Figure 18:*** Test Run 3 (RSA Cryptosystem)





***Figure 19:*** Test Run 4 (Quiz)



# **Discussion of Results**

According to the figures shown in the ‘Results’ section, four (4) test runs were performed in order to cover all components of the program. Among the test runs performed, three of them were primarily based on the simulation of the cryptographic methods namely the shift cipher, vigenere cipher, and the RSA cryptosystem. Within these test runs, encryption was first performed by inputting the necessary elements that established the algorithm of a particular cryptographic method. Next, decryption was performed by inputting or processing (in the case of the RSA cryptosystem) the encrypted text in order to ensure the functionality and accuracy of the encryption and decryption processes in each cryptographic method. Additionally, the test runs also included the modification of specific elements to establish a variety of outcomes such as the message, keyword, and key. Other than processes involving the cryptographic algorithms, the test runs also addressed instances wherein invalid inputs are situated in the program. Based on the three test runs, inputting an invalid number among the given prompts will endlessly redirect the user into the input segment until the user inputs a valid value. Aside from the test runs on the simulation of the cryptographic methods, the fourth and last test run highlights the quiz component. Within this test run, it covered every possible answer format as it is important in assuring the accuracy of the quiz’s assessment and grading. It includes answer inputs with whitespaces and answer inputs that are in lowercase (since the predefined correct answers are in uppercase). Lastly, the test runs were all terminated in the main menu, particularly by entering “3” as it signifies the “Exit” or termination of the program.

# **Analysis, Conclusion and Future Directives**

In conclusion, the project "Cryphytonology: An Integration of Python Programming in Message Encryption and Decryption" has successfully accomplished its primary objectives of integrating Python programming to simulate various cryptographic methods, such as the Shift cipher, Vigenere cipher, and RSA cryptosystem. These methods have their own unique algorithms and processes for encryption and decryption, which are fundamental in acquiring knowledge in the field of cryptography.

As a recommendation for future laboratory projects and experiments, it is advised to incorporate a wider range of cryptography methods, particularly those that are complex and contemporary, in order to create a more diverse and comprehensive learning platform for cryptography. Furthermore, the project suggests that other researchers continue to explore, study, and apply additional cryptographic methods, as these can be crucial in assessing cybersecurity and data privacy concerns.

Lastly, displaying comparisons between cryptographic methods based on their strengths and weaknesses, particularly in real-life applications, can be valuable in future projects and experiments. This can aid in determining the suitability of implementing specific cryptographic methods in different situations and conditions.

# References

*A brief history of encryption (and cryptography)*. (2023, February 1). Thales Group. <https://www.thalesgroup.com/en/markets/digital-identity-and-security/magazine/brief-history-encryption>

Boneh, D. (1999). Twenty years of attacks on the RSA cryptosystem. Ams.org. <https://www.ams.org/journals/notices/199902/boneh.pdf?trk=199902boneh&cat=collection>

Cisco Umbrella. (2022, August 10). *Cybersecurity threat trends report - Cisco Umbrella*. <https://umbrella.cisco.com/info/cybersecurity-threat-trends-report>

Difference between Monoalphabetic Cipher and Polyalphabetic Cipher. (2023). *BYJUS*. <https://byjus.com/gate/difference-between-monoalphabetic-cipher-and-polyalphabetic-cipher/>

Internet Society. (2022, August 24). What is TLS & How Does it Work? | ISOC Internet Society. https://www.internetsociety.org/deploy360/tls/basics/?gclid=CjwKCAjw6IiiBhAOEiwALNqncdvogDShFm6\_1YQ-YbLPOxYwgJLBQpqAGdc8DIwevHTxM\_I\_SgDhWxoCv-gQAvD\_BwE

Leighton, A. C. (1969). Secret Communication among the Greeks and Romans. *Technology and Culture*, *10*(2), 139–154. <https://doi.org/10.2307/3101474>

Nocon, R., Nocon, E. (2018). Essential Mathematics in the Modern World (2nd ed.). C & E Publishing Inc. p157-p185.

Olmstead, K., & Smith, A. (2022, March 8). What the Public Knows About Cybersecurity. *Pew Research Center: Internet, Science & Tech*. https://www.pewresearch.org/internet/2017/03/22/what-the-public-knows-about-cybersecurity/

SecureGuard: A Certificate Validation System in Public Key Infrastructure. (2018, June 1). IEEE Journals & Magazine | IEEE Xplore. <https://ieeexplore.ieee.org/document/8290970>

Simpson S. (1997). Cryptography in Everyday Life. <https://www.laits.utexas.edu/~anorman/BUS.FOR/course.mat/SSim/life.html>

Simmons, G. J. (2023, February 25). Vigenère cipher. Encyclopedia Britannica. <https://www.britannica.com/topic/Vigenere-cipher>

SSL Corp. (2021, October 13). What is HTTPS? - SSL.com. SSL.com. <https://www.ssl.com/faqs/what-is-https/>

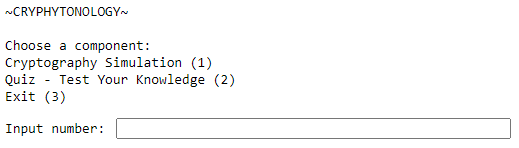
*The History of Cryptography*. (n.d.). <https://cs.stanford.edu/people/eroberts/courses/soco/projects/public-key-cryptography/history.html>

# 

# Appendices

## User’s Manual

**Main Menu Interface:**



Upon starting the program, the user will be greeted with the title of the program and a menu that allows them to choose a component they wish to enter. They will enter that component by inputting the number associated with that component.

1. Cryptography Simulation

* Upon inputting 1, the user will be sent to another menu that enables them to choose between 3 different cryptographic methods to apply.

1. Quiz - Test Your Knowledge

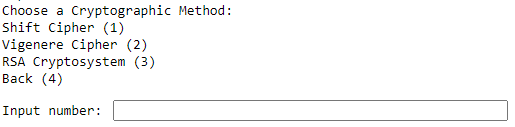
* Upon inputting 2, a quiz will begin that tests their knowledge surrounding Cryptography.

1. Exit

* Ends the execution of the program.

## 

**Cryptography Simulation Interface:**



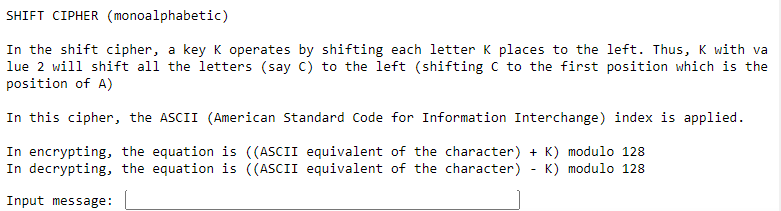
In the cryptography simulation menu, users will be able to apply different cryptographic methods. They will input the number that corresponds to the cryptographic method they would like to apply.

1. Shift Cipher
2. Vigenere Cipher
3. RSA Cryptosystem
4. Return

**Quiz Interface:**

* Questions will be shown to the user sequentially and they are to correctly identify what is being asked.
* The quiz consists of 5 questions and the number of questions correctly answered by the user is tracked and shown to them at the end of the quiz.
* Inputs are to be single worded and alphanumeric only.
* After the quiz, the user will be sent back to the main menu.

**Shift Cipher Interface:**



Upon inputting 1 in the cryptography simulation menu, the user will enter the shift cipher component where it will be explained to the user how shift ciphers work.

1. They will be asked for the message they would like to encrypt or decrypt using the shift cipher.

* Valid inputs are any characters within the ASCII index.

1. Then, they will be asked what value of *K* they would like to be used in encrypting or decrypting the message.

* Input must be integers only.

1. Next, the user will be asked if they would like to encrypt or decrypt the input message.

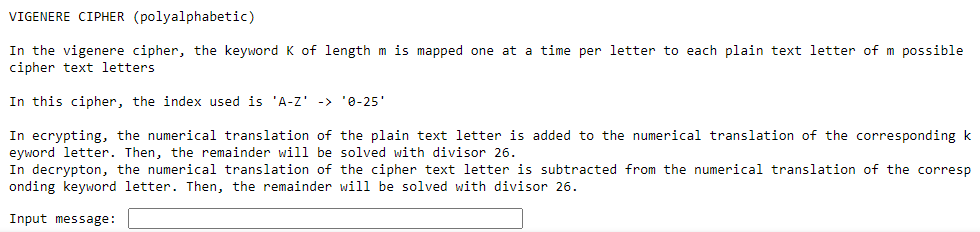
* Inputs must be integers only.

1. Upon inputting 1, the inputted message will be encrypted and shifted *K* values to the right.
2. Upon inputting 2, the inputted message will be decrypted and shifted *K* values to the left.
3. Finally, the resulting plaintext or ciphertext is shown to the user before being asked if they would like to change the message, change the value of *K*, or exit the component.

* Inputs must be integers only.

1. Upon inputting 1, the user will be sent back to step 1 and be able to input a different message to encrypt or decrypt.
2. Upon inputting 2, the user will be sent back to step 2 and change the value of *K* that the message will be encrypted or decrypted with.
3. Upon inputting 3, the user will be sent back to the cryptography simulation menu where they will be able to select a different cryptographic method to apply.

**Vigenere Cipher Interface:**



Upon inputting 2 in the cryptography simulation menu, the user will enter the Vigenere cipher component where it will be explained to the user how Vigenere ciphers work.

1. The user will be asked for the message they would like to encrypt or decrypt using the Vigenere cipher.

* Input must consist of the english alphabet only.

1. Then, the user will be asked for the keyword that they would like to use to encrypt or decrypt the input message.

* Input must consist of the english alphabet only.

1. Next, the user will be asked if they would like to encrypt or decrypt the message using the keyword provided.

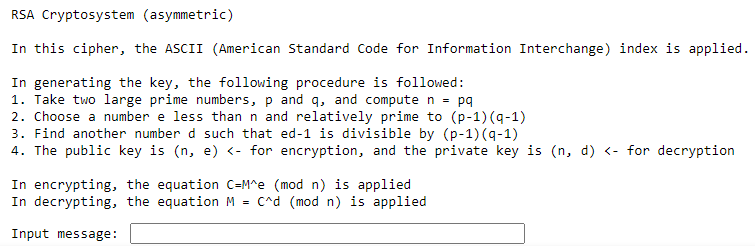
* Input must be integer only

1. Upon inputting 1, the input message will be encrypted using the code word.
2. Upon inputting 2, the input message will be decrypted using the code word.
3. Finally, the resulting plaintext or ciphertext is shown to the user before being asked if they would like to change the input message, change the keyword used, or exit out of the component.

* Input must be integer only

1. Upon inputting 1, the user will be sent back to step 1 and be able to choose a new message to encrypt or decrypt using the Vigenere cipher.
2. Upon inputting 2, the user will be sent back to step 2 where they will be able to select a new keyword to encrypt or decrypt the input message with.
3. Upon inputting 3, the user will be sent back to the cryptography simulation menu where they will be able to select a different cryptographic method to apply.

**RSA Cryptosystem Interface:**



Upon inputting 3 in the cryptography simulation menu, the user will enter the RSA cryptosystem component where it will be explained to the user how the RSA cryptosystem is used in encrypting and decrypting messages.

1. The user will be asked for the message they would like to encrypt or decrypt using the RSA cryptosystem.

* Valid inputs are any characters within the ASCII index.

1. Then, the user will be asked to input two different large prime numbers.

* Valid inputs are large prime integers only.

1. The resulting public key and private key will be shown to the user, along with the resulting encrypted text. The user is then asked if they would like to change the message, decrypt the message, or exit the component.

* Valid inputs are integers only.

1. Upon inputting 1, the user will be asked for the next message they would like to encrypt using the same public key.

* Valid inputs are any characters within the ASCII index.

1. Upon inputting 2, the resulting encrypted text will be decrypted using the private key. The user will then be asked if they would like to change the message or exit the component.
   1. Upon inputting 1, the user will be asked for the next message they would like to encrypt using the same public key before displaying its corresponding encryption and sending the user back to the beginning of step 3.
   2. Upon inputting 2, the user will be sent back to the cryptography simulation menu where they will be able to select a different cryptographic method to apply.

## 

## 

## Source Code

import math

def main():

# This module is a menu-based system that allows users to simulate the different given

# given cryptographic methods and take a quiz to test their knowledge. It continuously loops

# until the user chooses to exit the program. Users can select options such as cryptography simulation

#, quiz or exit. Depending on their choice certain statements will be performed.

# Choosing cryptography simulation will direct the user to another set of options containing the three cryptographic methods.

# Choosing quiz will direct the user to the quiz component which contains a set of questions in which the user needs to answer.

# Choosing exit will terminate the system. In the module, invalid options will force the user to input another value.

system\_run = True

while system\_run:

print("~CRYPHYTONOLOGY~\n\nChoose a component: \nCryptography Simulation (1) \nQuiz - Test Your Knowledge (2) \nExit (3)")

user\_choice = int(input("Input number: "))

if user\_choice == 1:

choose\_opt = True

while choose\_opt:

print("Choose a Cryptographic Method: \nShift Cipher (1) \nVigenere Cipher (2) \nRSA Cryptosystem (3) \nBack (4)")

user\_crypt = int(input("Input number: "))

if user\_crypt == 1:

shift()

elif user\_crypt == 2:

vigenere()

elif user\_crypt == 3:

rsa()

elif user\_crypt == 4:

break

else:

print("Invalid Option\n")

continue

elif user\_choice == 2:

quiz()

elif user\_choice == 3:

print("Thank you for using CRYPHYTONOLOGY!")

break

else:

print("Invalid Option\n")

continue

exit()

def shift():

# This module implements the interface for the shift cipher

print('\n')

print('SHIFT CIPHER (monoalphabetic)\n')

print('In the shift cipher, a key K operates by shifting each letter K places to the left. Thus, K with value 2 will shift all the letters (say C) to the left (shifting C to the first position which is the position of A)')

print('\nIn this cipher, the ASCII (American Standard Code for Information Interchange) index is applied. ')

print('\nIn encrypting, the equation is ((ASCII equivalent of the character) + K) modulo 128 ')

print('In decrypting, the equation is ((ASCII equivalent of the character) - K) modulo 128 ')

message = input("Input message: ")

key = int(input("Input key: "))

shift\_choice = 1

switch = 0

while shift\_choice == 1 or shift\_choice == 2:

if switch == 1:

message = input("Input message: ")

if switch == 2:

key = int(input("Input key: "))

choice\_opt1 = True

while choice\_opt1:

print("Encrypt (1) or Decrypt (2)")

enc\_or\_dec = int(input("Input number: "))

if enc\_or\_dec == 1:

print("The encrypted message is", shift\_enc(message, key))

break

elif enc\_or\_dec == 2:

print("The decrypted message is", shift\_dec(message, key))

break

else:

print("Invalid Option\n")

continue

choice\_opt2 = True

while choice\_opt2:

print("Change message? (1) \nChange key? (2) \nExit(3)")

shift\_choice = int(input("Input number: "))

if shift\_choice == 1:

switch = 1

break

elif shift\_choice == 2:

switch = 2

break

elif shift\_choice == 3:

return

else:

print("Invalid Option\n")

continue

def vigenere():

# This module implements the interface for the vigenere cipher

print('\n')

print("VIGENERE CIPHER (polyalphabetic)\n")

print("In the vigenere cipher, the keyword K of length m is mapped one at a time per letter to each plain text letter of m possible cipher text letters\n")

print("In this cipher, the index used is 'A-Z' -> '0-25'\n")

print("In encryption, the numerical translation of the plain text letter is added to the numerical translation of the corresponding keyword letter. Then, the remainder will be solved with divisor 26.")

print("In decryption, the numerical translation of the cipher text letter is subtracted from the numerical translation of the corresponding keyword letter. Then, the remainder will be solved with divisor 26.")

message = input("Input message: ").upper()

keyword = input("Input keyword: ").upper()

keys = vigenere\_key(message, keyword)

vigenere\_choice = 1

switch = 0

while vigenere\_choice == 1 or vigenere\_choice == 2:

if switch == 1:

message = input("Input message: ").upper()

keys = vigenere\_key(message, keyword)

if switch == 2:

keyword = input("Input keyword: ")

keys = vigenere\_key(message, keyword)

print("The modified keyword is ", keys)

choose\_opt1 = True

while choose\_opt1:

print("Encrypt (1) or Decrypt (2)")

enc\_or\_dec = int(input("Input number: "))

if enc\_or\_dec == 1:

print("The encrypted text is", vigenere\_enc(message, keys))

break

elif enc\_or\_dec == 2:

print("The decrypted text is", vigenere\_dec(message, keys))

break

else:

print("Invalid Option\n")

continue

choose\_opt2 = True

while choose\_opt2:

print("Change message? (1) \nChange keyword? (2) \nExit(3)")

vigenere\_choice = int(input("Input number: "))

if vigenere\_choice == 1:

switch = 1

break

elif vigenere\_choice == 2:

switch = 2

break

elif vigenere\_choice == 3:

return

else:

print("Invalid Option\n")

continue

def rsa():

# This module implements the interface for the rsa cryptosystem

print('\n')

print("RSA Cryptosystem (asymmetric)\n")

print('In this cipher, the ASCII (American Standard Code for Information Interchange) index is applied.\n')

print("In generating the key, the following procedure is followed:")

print("1. Take two large prime numbers, p and q, and compute n = pq")

print("2. Choose a number e less than n and relatively prime to (p-1)(q-1)")

print("3. Find another number d such that ed-1 is divisible by (p-1)(q-1)")

print("4. The public key is (n, e) <- for encryption, and the private key is (n, d) <- for decryption\n")

print("In encrypting, the equation C=M^e (mod n) is applied")

print("In decrypting, the equation M = C^d (mod n) is applied")

message = input("Input message: ")

user\_input = True

while user\_input:

num1 = int(input("Input a large prime number: "))

num2 = int(input("Input another large prime number: "))

is\_prime1 = prime\_check(num1)

is\_prime2 = prime\_check(num2)

repeat = False

if is\_prime1 == False and is\_prime2 == False:

print("Both inputs are not prime numbers\n")

repeat = True

elif is\_prime1 == False and is\_prime2 == True:

print("First Input is not a prime number")

repeat = True

elif is\_prime1 == True and is\_prime2 == False:

print("Second Input is not a prime number")

repeat = True

else:

key = prime\_key(num1, num2)

print("Your public key is ", key[0])

print("Your private key is ", key[1])

break

if repeat:

choice\_opt3 = True

while choice\_opt3:

print("Input again (1)? or Exit (2)?")

repeat\_choice = int(input("Input number: "))

if repeat\_choice == 1:

break

elif repeat\_choice == 2:

return

else:

print("Invalid Option\n")

continue

rsa\_choice = 1

switch = 0

while rsa\_choice == 1 or rsa\_choice == 2:

if switch == 1:

message = input("Input message: ")

if switch == 2:

decrypt = rsa\_dec(ciphertext, key)

print("The decrypted text is ", decrypt)

choice\_opt1 = True

while choice\_opt1:

print("Change message? (1) \nExit(2)")

rsa\_choice2 = int(input("Input number: "))

if rsa\_choice2 == 1:

switch = 1

break

elif rsa\_choice2 == 2:

return

else:

print("Invalid Option\n")

continue

if rsa\_choice2 == 1:

continue

ciphertext = rsa\_enc(message, key)

if switch != 2:

print("The encrypted text is ", ciphertext)

choice\_opt2 = True

while choice\_opt2:

print("Change message? (1) \nDecrypt the message? (2) \nExit(3)")

rsa\_choice = int(input("Input number: "))

if rsa\_choice == 1:

switch = 1

break

elif rsa\_choice == 2:

switch = 2

break

elif rsa\_choice == 3:

return

else:

print("Invalid Option\n")

continue

def quiz():

# This module presents a series of questions to the user and evaluates their answers based on the predefined set of correct answers. The total score obtained by the user will be displayed after completing the quiz

questionnaire = {"First" : {'Question' : 'What is the process of using an algorithm to transform information into a format that cannot be read?', "Answer" : "ENCRYPTION"}

, "Second" : {'Question' : 'Is Vigenere Cipher monoalphabetic or polyalphabetic?', "Answer" : "POLYALPHABETIC"}

, "Third" : {'Question' : 'If the encrypted message is PFCGTE from LBYCPA, what is the key of the shift cipher?', "Answer" : "4"}

, "Fourth" : {'Question' : 'What key is used to decrypt in RSA cryptosystem?', "Answer" : "PRIVATE"}

, "Fifth" : {'Question' : 'Which of the following cryptographic methods uses a keyword?', "Answer" : "VIGENERE"}

}

score = 0

for key in questionnaire:

print(questionnaire[key]['Question'])

answer = input().strip().upper()

if answer == questionnaire[key]['Answer']:

score += 1

print("Correct!")

else:

print("Incorrect!")

print("Quiz completed! You scored {}/{} points.".format(score, len(questionnaire)))

def shift\_enc(message, key):

# This module implements the encryption algorithm of the shift cipher

encrypt = ""

for char in message:

ascii\_equivalent = (ord(char) + key) % 128

encrypt += chr(ascii\_equivalent)

return encrypt

def shift\_dec(message, key):

# This module implements the decryption algorithm of the shift cipher

decrypt = ""

for char in message:

ascii\_equivalent = (ord(char) - key) % 128

decrypt += chr(ascii\_equivalent)

return decrypt

def vigenere\_enc(message, keyword):

# This module implements the encryption algorithm of the vigenere cipher

encrypt = ""

for i in range(len(message)):

alphabet\_num = (ord(message[i]) + ord(keyword[i])) % 26

encrypt += chr(alphabet\_num + ord('A'))

return encrypt

def vigenere\_dec(message, keyword):

# This module implements the decryption algorithm of the vigenere cipher

decrypt = ""

for i in range(len(message)):

alphabet\_num = (ord(message[i]) - ord(keyword[i])) % 26

decrypt += chr(alphabet\_num + ord('A'))

return decrypt

def rsa\_enc(message, pubkey):

# This module implements the encryption algorithm for the RSA cryptosystem.

encrypt = []

n = pubkey[0][0]

e = pubkey[0][1]

for char in message:

ascii\_equivalent = ord(char)

encrypted = pow(ascii\_equivalent, e, n)

encrypt.append(str(encrypted))

ciphertext = (' '.join(encrypt))

return ciphertext

def rsa\_dec(message, privkey):

# This module implements the decryption algorithm for the RSA cryptosystem.

decrypt = []

n = privkey[1][0]

d = privkey[1][1]

split\_values = message.split(' ')

for value in split\_values:

decrypted = pow(int(value), d, n)

decrypt.append(str(decrypted))

ciphertext = (' '.join(decrypt))

return ciphertext

def prime\_check(num):

# This module checks if the given number is a prime number or not.

is\_prime = True

if num <= 1:

is\_prime = False

else:

for i in range(2, num):

if num % i == 0:

is\_prime = False

break

return is\_prime

def prime\_key (num1, num2):

# This module generates the public and private keys for the RSA cryptosystem using the two number inputs num1 and num2

n = num1 \* num2

r = (num1 - 1) \* (num2 - 1)

e = 2

while e < r:

if (math.gcd(e, r) == 1):

break

else:

e += 1

d = pow(e, -1, r)

result = ([n, e], [n,d])

return result

## Work Breakdown

| **Name** | **Tasks Assigned** | **Percentage of the Work Contribution** |
| --- | --- | --- |
| **Miguel Martinez** | Review of Related Literature  Introduction - Background of the Study  User Manual  References | **27%** |
| **Sean Morales** | Methodology - Flowchart  Methodology - Pseudocode  Source Code of the Program (All the modules)  Introduction - Objectives  Introduction - Significance of the Study  Introduction - Problem Statement  Results and Discussion  Overlooking the Groups’ Progress | **46%** |
| **Nicole Salonga** | Methodology - IPO Chart  Methodology - Pseudocode (*comments)*  Source Code of the Program (Organizing the modules, comments)  Main Graphics Designer for Visual Aids (Poster, Timetable of Activities)  Organizing and Proofreading Documents  Analysis, Conclusion, and Future Directives | **27%** |

## Personal Data Sheet

**Personal Data Sheet #1**

**Full Name:** Sean Alexander M. Morales

**Nickname:** Sean

**Birthday:** October 31, 2003

**Age:** 19

**Contact Email:** [sean\_morales@dlsu.edu.ph](mailto:sean_morales@dlsu.edu.ph)

**College:** Gokongwei College of Engineering (GCOE)

**Program:** Bachelor of Science in Computer Engineering

**Personal Data Sheet #2**

**Full Name:** Princess Nicole Salonga

**Nickname:** Nicole

**Birthday:** November 14, 2003

**Age:** 19

**Contact Email:** princess\_salonga@dlsu.edu.ph

**College:** Gokongwei College of Engineering (GCOE)

**Program:** Bachelor of Science in Computer Engineering

**Personal Data Sheet #3**

**Full Name:** Miguel James R. Martinez

**Nickname:** Emjay

**Birthday:** July 2, 2004

**Age:** 18

**Contact Email:** miguel\_martinez@dlsu.edu.ph

**College:** Gokongwei College of Engineering (GCOE)

**Program:** Bachelor of Science in Computer Engineering