ISO 9001:2015
TÜV Rheinland ID 910863351

CS103-Computer Programming 2
Programming Project
Project Documentation

I.1.A) Introduction

State here the purpose of the system/program that you are going to make, who will be the intended users and how it will benefit them. Provide a detailed description of the features/functions available in your system.

It is essential to prioritize secure communication in the current digital era. As the internet becomes more available and more aspects of our lives go digital, safeguarding our privacy in conversations is now more crucial than ever. This program explores the importance of secure communication and highlights algorithms such as the RSA encryption technology and the complex math behind cryptography.

Our program, the RSA Cipher Tool, is a tool that secures messages using the RSA encryption algorithm. RSA is a form of asymmetric cryptography, requiring public and private keys for encryption and decryption. Users have the capability to create both public and private keys, which allows them to safely store and send messages and guarantee that only the specified recipient can decode the encrypted text.

Due to C's data type limitations and how cryptography deals with really large numbers, a custom Bignum library was made to handle the operation and arithmetic of bignums.

The RSA Cipher Tool was made to be accessible, easy to use, and work consistently across all devices. The. Users are given 4 main options:

1. Generate Keys: The user can choose between different key sizes, with each increasing key size, the security increases but the encrypt/decrypt speed decreases.
2. Encrypt Data: The user can use the generated keys to encrypt a text file they want to keep secret, and even encrypt text they want to send to someone using the recipient's public key.
3. Decrypt Data: The user can then decrypt the message that was encrypted using their public key and reveal the plaintext.
4. About: A section that covers the story behind the project and a quick guide on how to operate it.

All menu options output and receive data in a similar way for uniformity. Details related to the action being done are also given to the user and a loading bar and status. Each option was meticulously crafted to be as easy to use, hassle free, and secure as possible.

In conclusion, the RSA Cipher Tool provides a thorough method for ensuring secure communication, giving users strong encryption options. In today's world, the RSA Cipher Tool is a crucial component of digital privacy and security, thanks to its use of strong encryption algorithms, secure key management systems, and seamless integration with secure communication protocols.

I.1.B) Algorithm

Provide a pseudocode of the main steps of the program.

```
Clear the screen
Get the terminal size
Print program header
Get the option from user and store in userMenuState to determine what action to perform
Repeat the following until the userMenuState is a valid option
  If the userMenuState is a valid option, then
        Perform the corresponding action:
                If chosen userMenuState == 1
                        Then, generate public and private keys with the chosen bit size of the user
                Else if chosen userMenuState == 2
                        Then, encrypt the content of the entered file using the given public key
                Else if chosen userMenuState == 3
                        Then, decrypt the content of the entered file using the user’s private key
                Else if chosen userMenuState == 4
                        Then, display the description of of the RSA Cipher Tool program
                Else if chosen userMenuState == 5
                        Then, clear the screen and output “Exiting RSA Cipher Tool…” and deallocate all
                        the bignums used.
                        End the program
  Else
        Clear the input
        Output “Invalid number!”
        Output “Please choose a number between 1 - 5.”
```

I.1.C) Program Specification

A. Standard C Functions used

Identify the predefined functions with their header files that were used in the program.

Standard C Functions used in rsa-cipher-tool.c

HEADER FILES	PREDEFINED FUNCTIONS	PURPOSE
stdio.h	`printf();`	to print formatted output to the console or to a file.
	`scanf();`	to read formatted input from the console or from a file.
	`fprintf();`	to write formatted output to a file.
	`fscanf();`	to read formatted input from a file.
	`fclose();`	to close a file that was previously opened with fopen() or freopen().
	`fopen();`	to open a file.
	`fgets();`	to read a line of text from a file stream or a standard input and store it in a string variable.
	`fseek();`	to move the file pointer associated with a given file to a specific position
	`rewind();`	to set the file position to the beginning of the file for the stream pointed to by the given file pointer
	`fgetc();`	to read the next character from the file associated with the passed file pointer
	`getchar();`	to read the next character from the standard input (stdin)
string.h	`strcmp();`	to compare two strings and determine if they are equal or not.
	`strcpy();`	to copy a string from one memory location to another
	`strlen();`	to determine the length of a string.
	`strtok();`	to tokenize a string into a sequence of tokens
	`strcat():`	to concatenate two strings.
	`strcspn();`	to find the length of the initial segment of a string that does not contain any characters from a given set of characters.
unistd.h	`sleep();`	to suspend the execution of the current thread for a specified number of seconds.
ctype.h	`tolower();`	to convert characters to lowercase.
	`isdigit();`	to check if a character is a digit or not
time.h	`clock();`	to return the processor time consumed by the program.
math.h	`log2();`	to return the binary logarithm (base-2 logarithm) of a number.
	`ceil();`	to round up a floating-point number to the nearest integer
stdlib.h	`exit();`	to terminate the program.

termios.h	`tcgetattr();`
	`tcsetattr();`
	`cfmakeraw();`
	`tcflush();`
	`tcsendbreak();`
	`tcdrain();`
		to get the cursor position if the program is not compiled on Windows API

Standard C Functions used in bignum.c		
HEADER FILES	PREDEFINED FUNCTIONS	PURPOSE
stdio.h	`printf();`	to print formatted output to the console or to a file.
	`scanf();`	to read formatted input from the console or from a file.
	`fprintf();`	to write formatted output to a file.
string.h	`memset();`	to fill a block of memory with a particular value
	`memcpy();`	to copy a block of memory from one location to another
	`strlen();`	to determine the length of a string.
stdlib.h	`malloc()`	dynamically allocates a block of memory at run-time and returns a pointer to the first byte of the allocated memory.
	`exit();`	to terminate the program.
	`calloc();`	to dynamically allocate memory
	`free();`	to deallocate memory that was previously allocated
	`srand();`	to seed the pseudo-random number generator
	`rand();`	to generate a pseudo-random number
time.h	`time();`	to get the current calendar time
math.h	`log2();`	to return the binary logarithm (base-2 logarithm) of a number.
	`log10();`	to compute the base-10 logarithm of a number
	`fmax();`	to find the maximum of two floating-point numbers
limits.h	`LLONG_MAX`	to get the maximum value of a long long int

B. Functions

List and provide detailed description in terms of its purpose, the parameters used, return values, local variables and operations of the user-defined functions.

Functions used in rsa-cipher-tool.c

1. generateKeys():
Purpose: This function generates RSA public and private keys.
Parameters: None.
Return Value: None.
Local Variables:

- KeySize keySizeOptions[]; (struct KeySize)
- int chosenKeySize, pPrivateLength, qPrivateLength, ePublicLength;
- Bignum nPublic, ePublic, dPrivate, one, pPrimePrivate, qPrimePrivate, phiOfNPrivate, pPrimePrivateMinusOne, qPrimePrivateMinusOne, plainChar, encryptedChar, decryptedChar;

Operations:

- Displays key size options and prompts users to choose a key size.
- Generates Keys.
- Tests generated keys.
- Outputs the generated keys.

2. encryptText():
Purpose: Encrypts text using RSA encryption.
Parameters: None.
Return Value: None.
Local Variables:

- FILE inputFilePtr, outputFilePtr;
- char inputFilename[], outputFilename[];
- Bignum nPublic, ePublic, encryptedChar, plainChar;

Operations:

- Prompts user for input file name..
- Retrieves public key.
- Encrypts each character of the input file and prints to the output file.

3. decryptText():
Purpose: Decrypts text encrypted with RSA encryption.
Parameters: None.
Return Value: None.
Local Variables:

- FILE inputFilePtr, outputFilePtr;
- char inputFilename[], outputFilename[];
- Bignum nPublic, dPrivate, encryptedChar,decryptedChar;

Operations:

- Prompts user for input file name..
- Retrieves private key.
- Decrypts each character of the input file and prints to the output file.

4. isValidEncryptedFile():
Purpose: Checks if a file is a valid RSA-encrypted file.
Parameters: FILE inputFilePtr
Return Value: boolean (1 for valid, 0 for invalid).
Operations: Checks if the file is a valid encrypted file by the program.

5. getInputFile():
Purpose: Prompts user for input file name and opens the file.
Parameters:

- FILE inputFilePtr;
- char inputFilenametype[]
- Action type; (enum Action)

Return Value: None.
Local Variables:

- char errorMessage[];

Operations:

- Prompts user for input file name until a valid file is provided.
- Opens the file
- Verifies if valid file

6. getKeys():
Purpose: Prompts user for RSA encryption/decryption keys.
Parameters:

- Action type; (enum Action)
- Bignum *ePublicOrDPrivate, *nPublic;

Return Value: None.
Local Variables: char errorPrompt[];
Operations: Prompts users for RSA keys until valid keys are provided characters.

7. about():
Purpose: Displays information about the program, including its purpose and contributors.
Parameters: None.
Return Value: None.
Local Variables: None.
Operations:

- Formats paragraphs using splitString().
- Displays formatted text about the program.
- Waits for user input before returning to the main menu.

8.splitString():
Purpose: Splits a string into substrings with a maximum line length specified by lineCap. It is used in the about() function to format paragraphs.
Parameters:

- char inputStr[];
- char *outputArr[];
- int *outputArrCount
- int lineCap;

Return Value: None.
Local Variables: None.
Operations: Splits the input string into substrings with a maximum length of characters to properly format the paragraphs when printing.

9. calculateLeftPadding():
Purpose: Calculates the left padding needed to center a string horizontally on the terminal screen.
Parameters: int strLength;
Return Value: int leftPadding;
Local Variables: None.
Operations: Calculates the left padding needed to center the string based on the terminal window size.

10. clearLines():
Purpose: Clears the content of multiple lines on the terminal screen, specified by startLine and endLine.
Parameters:

- int startLine, endLine;

Return Value: None.
Local Variables: None.
Operations: Clears the content of lines between startLine and endLine on the terminal screen.

11. clearPrompts():
Purpose: Clears the screen and prints the program header, essentially clearing the area under the program header.
Parameters: None.
Return Value: None.
Local Variables: None.
Operations: Clears the entire terminal screen and prints the program header.

12. clearScreen():
Purpose: Clears the entire terminal screen.
Parameters: None.
Return Value: None.
Local Variables: None.
Operations: Clears the entire terminal screen.

13. clearWord()
Purpose: Clears a specific word in a given line of the terminal screen.
Parameters:int y, Line number, startCo, endCol;
Return Value: None.
Local Variables: None.
Operations: Clears the content of the specified word in the given line on the terminal screen.

14. getCursorPosition():
Purpose: Retrieves the current cursor position on the terminal screen.
Parameters:int *x, *y;
Return Value: None.
Local Variables: None.
Operations: Retrieves the current cursor position on the terminal screen and stores the values in x and y.

15. getTerminalSize():
Purpose: Retrieves the size of the terminal window.
Parameters: None.
Return Value: None.
Local Variables: None.
Operations: Retrieves the size of the terminal window and returns the width and height.

16. hideCursor() and showCursor():
Purpose: Hide and show the cursor on the terminal screen, respectively.
Parameters: None.
Return Value: None.
Local Variables: None.
Operations: Hides or shows the cursor on the terminal screen.

17. loadingBar():
Purpose: Display a loading bar and status message at specified coordinates on the terminal screen, used to indicate progress during operations.
Parameters:int x, y, percentageDone;
Return Value: None.
Local Variables: None.
Operations: Display a loading bar with the specified percentage of progress and a status message at the specified coordinates on the terminal screen.

18. moveCursor():
Purpose: Moves the cursor to the specified coordinates on the terminal screen.
Parameters:int x, y;
Return Value: None.
Local Variables: None.
Operations: Moves the cursor to the specified coordinates on the terminal screen.

19. printProgramHeader():
Purpose: Prints the header of the program, typically displayed at the top of the screen.
Parameters: None.
Return Value: None.
Local Variables: None.
Operations: Prints the header of the program at the top of the screen.

20. promptExitConfirm():
Purpose: Prompts the user to confirm their intention to exit the program.
Parameters: None.
Return Value: None.
Local Variables: None.
Operations: Prompts the user to confirm whether they want to exit the program.

21. sleepProgram(int milliseconds):
Purpose: Suspends program execution for the specified number of milliseconds.
Parameters: milliseconds
Return Value: None.
Local Variables: None.
Operations: Suspends program execution for the specified number of milliseconds.

Functions used in bignum.c

1. int getLengthOfInteger(long long int num)
Purpose: Count the number of digits of positive or negative integer using log based solution
Parameters: long long int num
Return Value: Int
Local Variables: None.
Operations: Count the number of digits of an integer

2. void initBignum(Bignum *num);
Purpose: Initialize a Bignum to its default values and allocate an array for Bignum.digits[].
Parameters: (Bignum *num)
Return Value: None.
Local Variables:

- BignumNode *bignumNode (linked list struct)
- int *digitsPtr (pointer that was calloc'd)

Operations: Allocate memory for Bignum.digits[] using calloc() to set all indexes of the array to 0 , Set Bignum members to its default values, Add address of the Bignum

3. void freeAllBignums();
Purpose: Deallocate all the allocated Bignum.digits[] and the node all at once
Parameters: None.
Return Value: None.
Local Variables: None.
Operations: Iterate through the nodes starting from the head, and free all nodes in the list

4. void freeBignum();
Purpose: Deallocate a specified Bignum
Parameters: Bignum *num
Return Value: None.
Local Variables: None.
Operations: Free the dynamically allocated Bignum.digits[] and its corresponding linked list node

5. void printBignumNodeList();
Purpose: Prints the current list of Bignums that have not yet been freed. This is used for debugging memory leaks.
Parameters: None
Return Value: None.
Local Variables: None.
Operations: Loop through the linked list starting from the head, print its address.

6. void setBignum(Bignum * numStruct, char numStr[], BIGNUM_SIGN sign)
Purpose: Initialize a Bignum
Parameters: (Bignum * numStruct, char numStr[], BIGNUM_SIGN sign)
Return Value: None.
Local Variables: None.
Operations: Iterate through the string, load each digit character into the array.

7. void intToBignum(Bignum *numStruct, unsigned long long int integer, BIGNUM_SIGN sign);
Purpose: Convert the passed integer to Bignum
Parameters: (Bignum *numStruct, unsigned long long int integer, BIGNUM_SIGN sign)
Return Value: None.
Local Variables: int count
Operations: get the last digit of the integer, and load it to Bignum.digits[]

8. long long int bignumToInt(Bignum *num);
Purpose: Convert a Bignum to a long long integer
Parameters: (Bignum *num)
Return Value: long long int.
Local Variables:

- int maxNumOfDigits
- Long long int result
- Long long int multiplier

Operations: Multiply each digit to multiplier wherein each iteration multiplier is multiplied to 10, Add all numbers

9. int resetBignum(Bignum *num);
Purpose: Resets a Bignum to its initial state. This is useful as using this is more efficient than allocating another Bignum
Parameters: (Bignum *num)
Return Value: None.
Local Variables: None.
Operations: Go through each member and reset it to the default value. Use memset to set the digits of Bignum.digits[] to 0

10. void copyBignum(Bignum *result, Bignum *num);
Purpose: Copies a Bignum to another Bignum
Parameters: (Bignum *result, Bignum *num)
Return Value: None.
Local Variables: None.
Operations: Go through each member and copy it to result Bignum. memcpy Bignum.digits[] to the result Bignum.

11. void printBignum(Bignum *num);
Purpose: Prints the Bignum struct. (Bignum.digits[])
Parameters: (Bignum *num)
Return Value: None.
Local Variables: None.
Operations: Loop through Bignum.digits[] and print each digit

12. int isGreaterThanBignum(Bignum *num1, Bignum *num2);
Purpose: Compares two Bignums if one is greater than the other
Parameters: (Bignum *num1, Bignum *num2)
Return Value: boolean
Local Variables: None.
Operations: Loop through each digit looking for a difference. If the digits are different, return 0;

13. int isLessThanBignum(Bignum *num1, Bignum *num2);
Purpose: Compares two Bignums if one is less than the other
Parameters: (Bignum *num1, Bignum *num2)
Return Value: boolean
Local Variables: None.
Operations: Loop through each digit looking for a difference. If the digits are different, return 0;

14. int isEqualToBignum(Bignum *num1, Bignum *num2);
Purpose: Compares two Bignums if they are equal
Parameters: (Bignum *num1, Bignum *num2)
Return Value: boolean
Local Variables: None.
Operations: Loop through each digit looking for a difference. If the digits are different, return 0;

15. void addBignum(Bignum *result, Bignum *addend1, Bignum *addend2);
Purpose: Adds two Bignums
Parameters: (Bignum *result, Bignum *addend1, Bignum *addend2)
Return Value: None.
Local Variables: None.
Operations: Uses simple addition to add. Go through each digit starting from the LSD, and add its adjacent digit of the other Bignum.

16. void subtractBignum(Bignum *result, Bignum *minuend, Bignum *subtrahend);
Purpose: Subtracts two Bignums
Parameters: (Bignum *result, Bignum *minuend, Bignum *subtrahend)
Return Value: None.
Local Variables: None.
Operations: Uses simple subtraction to subtract. Go through each digit starting from the LSD, and subtract its adjacent digit of the other Bignum.

17. int multiplyBignum(Bignum *result, Bignum *multiplicand, Bignum *multiplier);
Purpose: Multiplies two Bignums
Parameters: (Bignum *result, Bignum *multiplicand, Bignum *multiplier)
Return Value: None.
Local Variables: None.
Operations: Uses the karatsuba multiplication algorithm to find the product of two bignums.

18. int divideBignum(Bignum *result, Bignum *dividend, Bignum *divisor);
Purpose: Divides two Bignums
Parameters: (Bignum *result, Bignum *dividend, Bignum *divisor)
Return Value: None.
Local Variables: None.
Operations: Use binary search to look for the quotient, once the quotient is found, return it.

19. int moduloBignum(Bignum *result, Bignum *dividend, Bignum *divisor);
Purpose: Find the modulo of a Bignum
Parameters: (Bignum *result, Bignum *dividend, Bignum *divisor)
Return Value: None.
Local Variables: None.
Operations: Use binary search to look for the quotient, once the quotient is found, return the difference between the dividend and (quotient * divisor)

20. int powerBignum(Bignum *result, Bignum *base, Bignum *exponent);
Purpose: Get base ^ exponent
Parameters: (Bignum *result, Bignum *base, Bignum *exponent)
Return Value: None.
Local Variables: None.
Operations: Turn the exponent into binary, iterate through each bit, and calculate the power.

21. int modularExponentiationBignum(Bignum *result, Bignum *base, Bignum *exponent, Bignum *divisor);
Purpose: Get the modular exponentiation
Parameters: (Bignum *result, Bignum *base, Bignum *exponent, Bignum *divisor)
Return Value: None.
Local Variables: None.
Operations: calculate (base^exponent) mod divisor

22. int modularInverseBignum(Bignum *result, Bignum *num, Bignum *divisor);
Purpose: Get the mod inverse
Parameters: (Bignum *result, Bignum *num, Bignum *divisor)
Return Value: None.
Local Variables: None.
Operations: find x in: (num * x) mod divisor = 1

23. int halfBignum(Bignum *result, Bignum *num);
Purpose: Divide a Bignum by 2 without using divideBignum()
Parameters: (Bignum *result, Bignum *num)
Return Value: int
Local Variables:

- int tempResultDigits[DEFAULT_BIGNUM_LENGTH]
- unsigned long long int resultLength
- int carry

Operations: Iterate through Bignum.digits[] and uses the native division on each digit

24. int generatePrimeBignum(Bignum *result, unsigned long long int primeLength);
Purpose: Generate a print Bignum given a desired length
Parameters: (Bignum *result, unsigned long long int primeLength)
Return Value: int
Local Variables:

- Bignum n
- Int primeLastDigits
- Int randPrimeLastDigitIndex
- Int isPrime

Operations: Generate random Bignum using the specified length, Test for its primality using the Miller Rabin Primality Test.

.

I.1.D) User Manual

Provide a detailed manual on how the program should be executed.

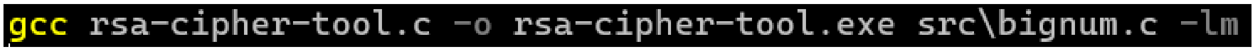
The RSA Cipher Tool is an encryption and decryption program developed for secure communication. It employs RSA encryption, a widely-used asymmetric cryptography algorithm, to protect sensitive data. The tool is also being implemented with a custom bignum library to handle the computation of really large numbers. Users can generate their own public and private keys, encrypt, and decrypt text. This tool aims to provide users with a reliable and user-friendly solution for handling secret messages using the RSA encryption algorithm.

User Manual for RSA Cipher Tool

I. **Running the program:**

To run the program, the user can open the executable file of the program.

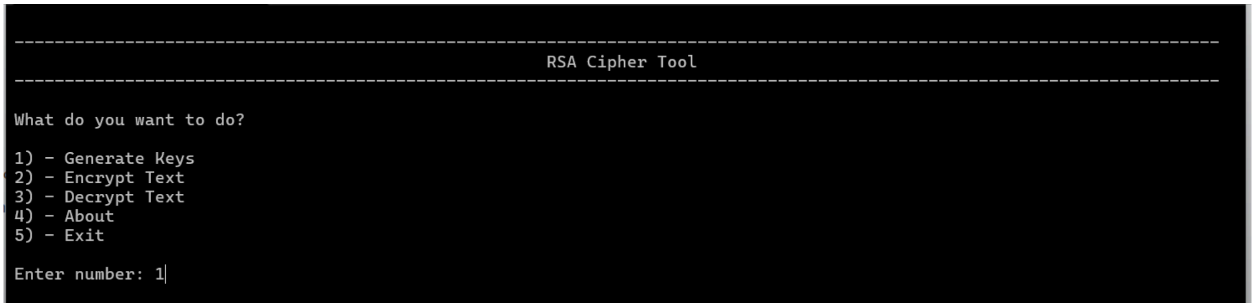
Compile the code using the command below. Since the program uses a custom header file, “src\bignum.c,” which is the path to the library file with respect to the executable file, was included. Additionally, the program uses math.h, hence, needing the -lm flag when compiling. Run the program using “rsa-cipher-tool.exe”



```
gcc rsa-cipher-tool.c -o rsa-cipher-tool.exe src\bignum.c -lm
```

II. **Usage:**

Upon running the program, the user is given the following options:

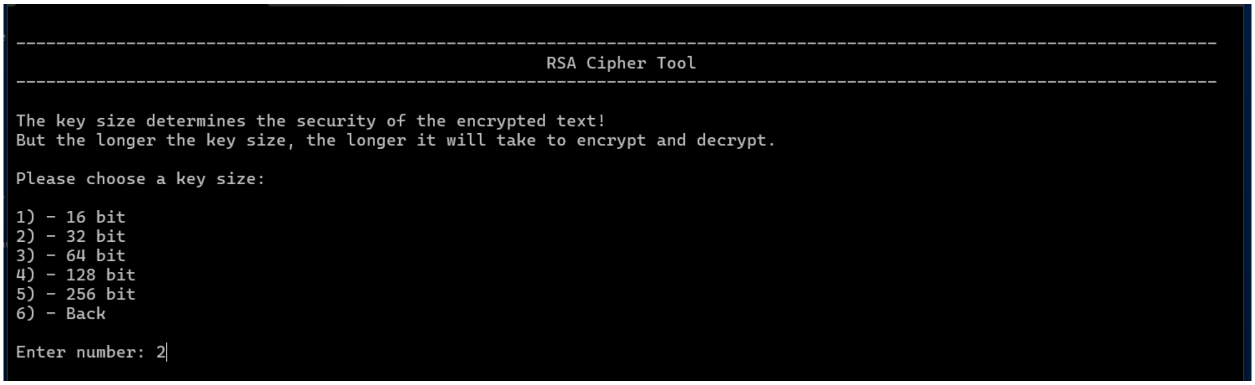


```
--------------------------------------------------------------------------------------------------------------------------
                                                 RSA Cipher Tool
--------------------------------------------------------------------------------------------------------------------------

What do you want to do?

1) - Generate Keys
2) - Encrypt Text
3) - Decrypt Text
4) - About
5) - Exit

Enter number: 1
```

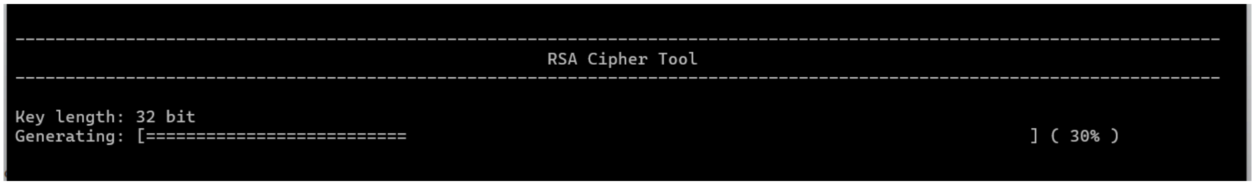
The user must choose from the valid range to continue. An invalid input will result in the program to iterate in the loop with an error message.

1) – Generate Keys

This option allows the user to generate their public and private keys for them to be able to encrypt and decrypt messages. The program asks the user for their preferred bit size. The higher key size, the more secure the keys are and the longer it will take for the program to generate their keys.

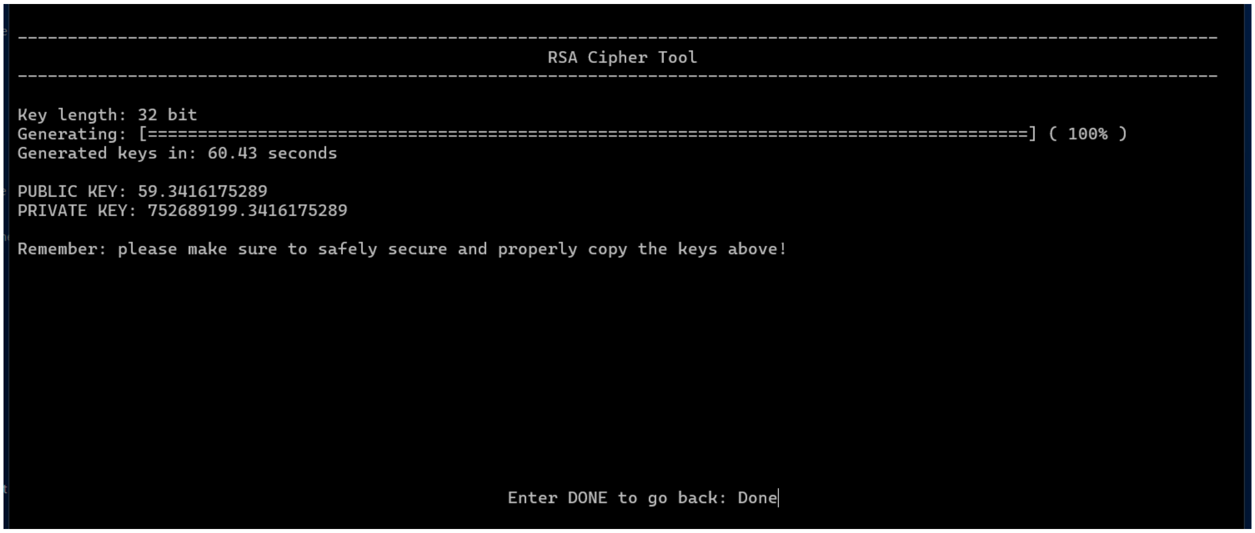


```
------------------------------------------------------------------------------------------------------------------------
                                                     RSA Cipher Tool
------------------------------------------------------------------------------------------------------------------------

The key size determines the security of the encrypted text!
But the longer the key size, the longer it will take to encrypt and decrypt.

Please choose a key size:

1) - 16 bit
2) - 32 bit
3) - 64 bit
4) - 128 bit
5) - 256 bit
6) - Back

Enter number: 2
```



```
------------------------------------------------------------------------------------------------------------------------
                                                     RSA Cipher Tool
------------------------------------------------------------------------------------------------------------------------

Key length: 32 bit
Generating: [==========================                                                              ] ( 30% )
```

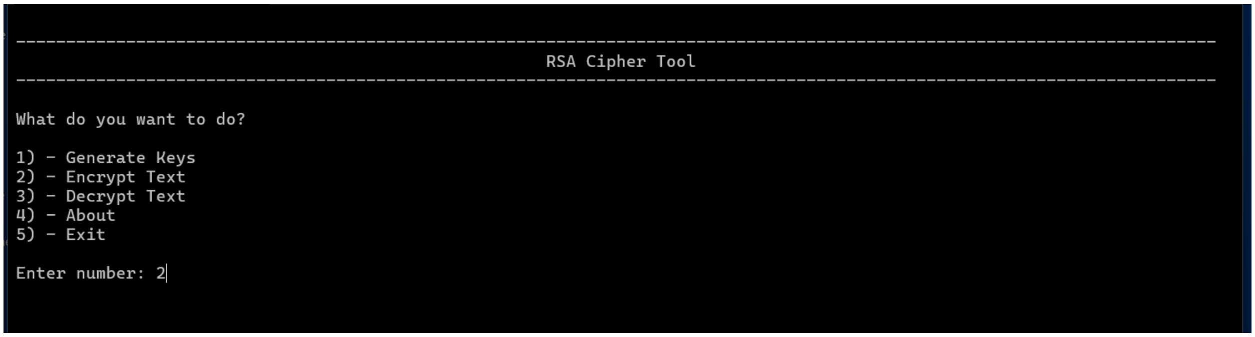
In all conditions or menu states, there is an exit prompt at the bottom of the terminal to back from the current state. This avoids accidental exits from the user and a confirmation that the user has finished copying the keys or transferring the encrypted/decrypted message.



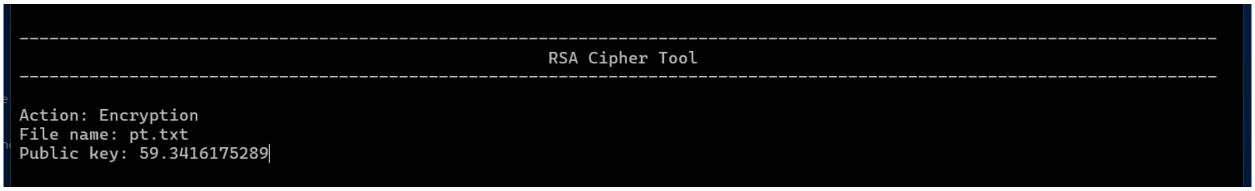
```
------------------------------------------------------------------------------------------------------------------------
                                                     RSA Cipher Tool
------------------------------------------------------------------------------------------------------------------------

Key length: 32 bit
Generating: [=========================================================================================] ( 100% )
Generated keys in: 60.43 seconds

PUBLIC KEY: 59.3416175289
PRIVATE KEY: 752689199.3416175289

Remember: please make sure to safely secure and properly copy the keys above!











                                                 Enter DONE to go back: Done
```

2) – Encrypt text

This option allows the user to encrypt the text by inputting the file name and the public key of the recipient of the message.

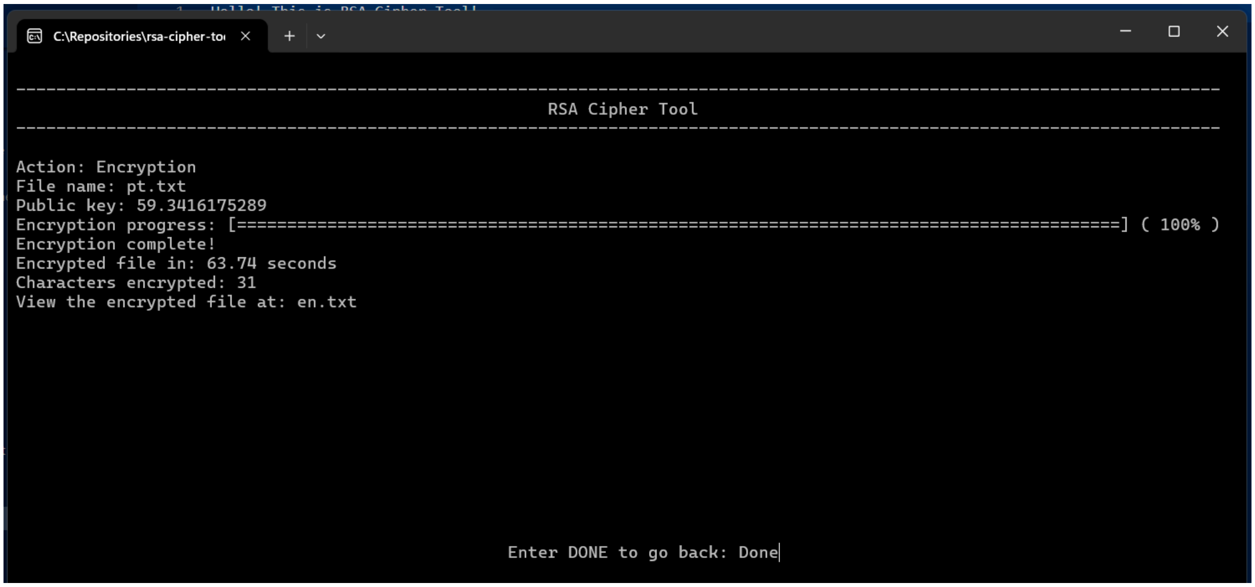


```
--------------------------------------------------------------------------------------------------------------------------
                                                     RSA Cipher Tool
--------------------------------------------------------------------------------------------------------------------------

What do you want to do?

1) - Generate Keys
2) - Encrypt Text
3) - Decrypt Text
4) - About
5) - Exit

Enter number: 2
```



```
--------------------------------------------------------------------------------------------------------------------------
                                                     RSA Cipher Tool
--------------------------------------------------------------------------------------------------------------------------

Action: Encryption
File name: pt.txt
Public key: 59.3416175289
```

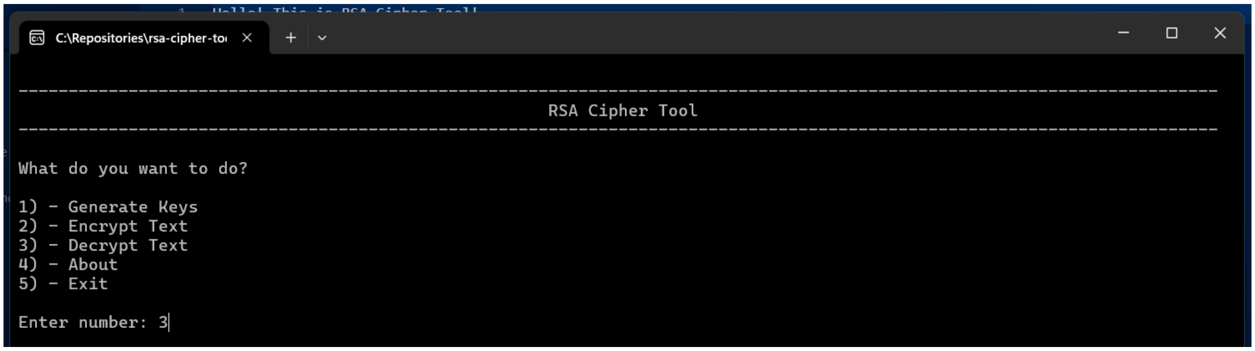
It will create the file “en.txt” and output the encrypted message. The resulting encrypted character will be an integer, thus, the encrypted characters are concatenated together and separated by a flag. In the terminal, it will display the encryption progress (loading bar and status), how long the encryption process was, number of characters encrypted, and where the user can find the encrypted message.



```
--------------------------------------------------------------------------------------------------------------------------
                                                     RSA Cipher Tool
--------------------------------------------------------------------------------------------------------------------------

Action: Encryption
File name: pt.txt
Public key: 59.3416175289
Encryption progress: [==========================================================================================] ( 100% )
Encryption complete!
Encrypted file in: 63.74 seconds
Characters encrypted: 31
View the encrypted file at: en.txt









                                                 Enter DONE to go back: Done
```

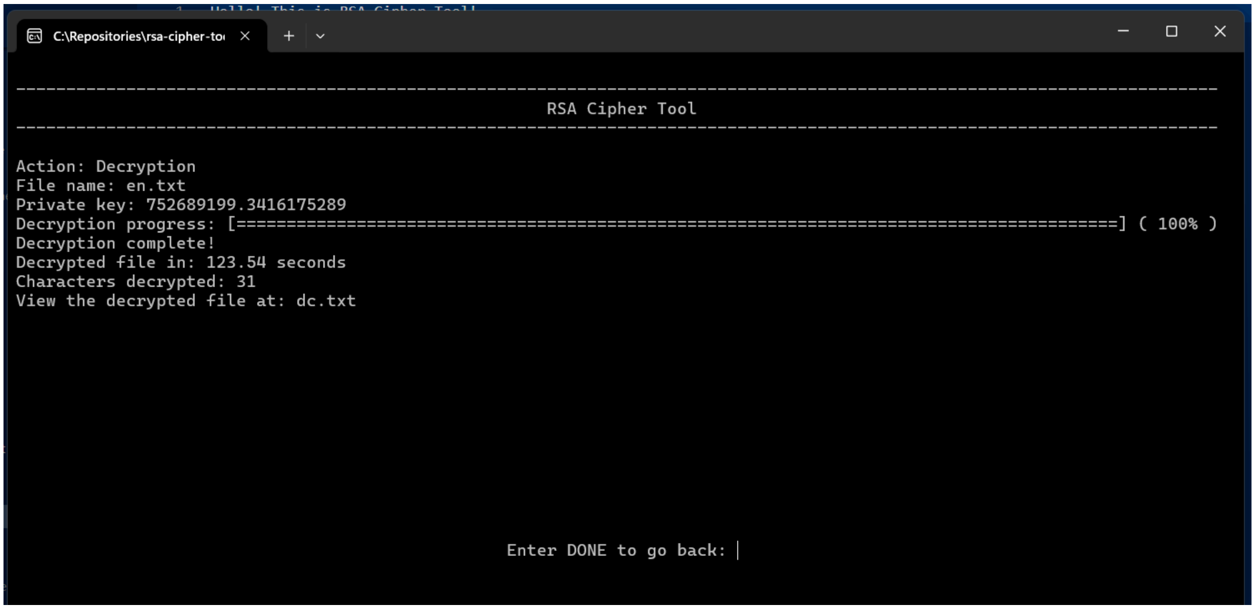
3) Decrypt text

This option is similar to the 2nd encryption option. Where the user will have to input the file name and private key to decrypt the file.



```
------------------------------------------------------------------------------------------------------------------------
                                                     RSA Cipher Tool
------------------------------------------------------------------------------------------------------------------------

What do you want to do?

1) - Generate Keys
2) - Encrypt Text
3) - Decrypt Text
4) - About
5) - Exit

Enter number: 3
```

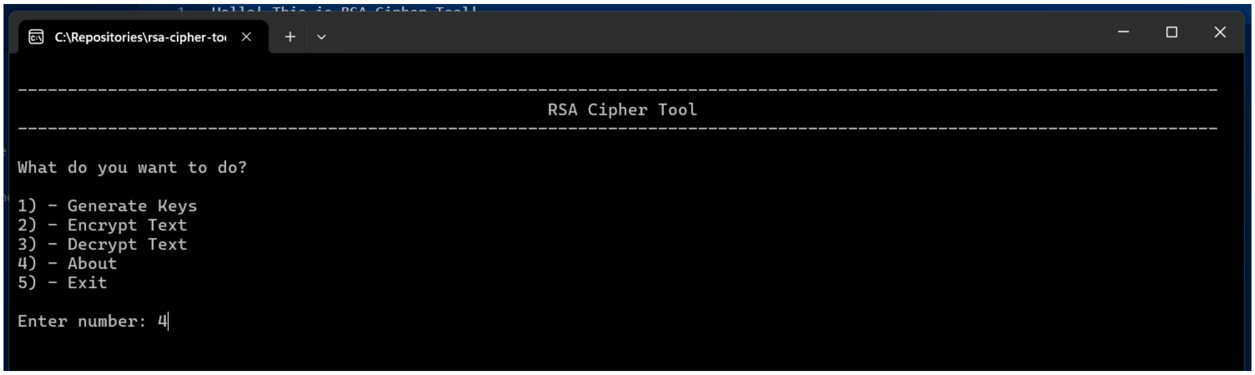
But, the input file for this decryption section will be first verified to check if the file that the user wants to decrypt is a valid file that was encrypted by the program. This ensures that the file can be properly decrypted. The output of this section is also similar to the 2nd encryption option.



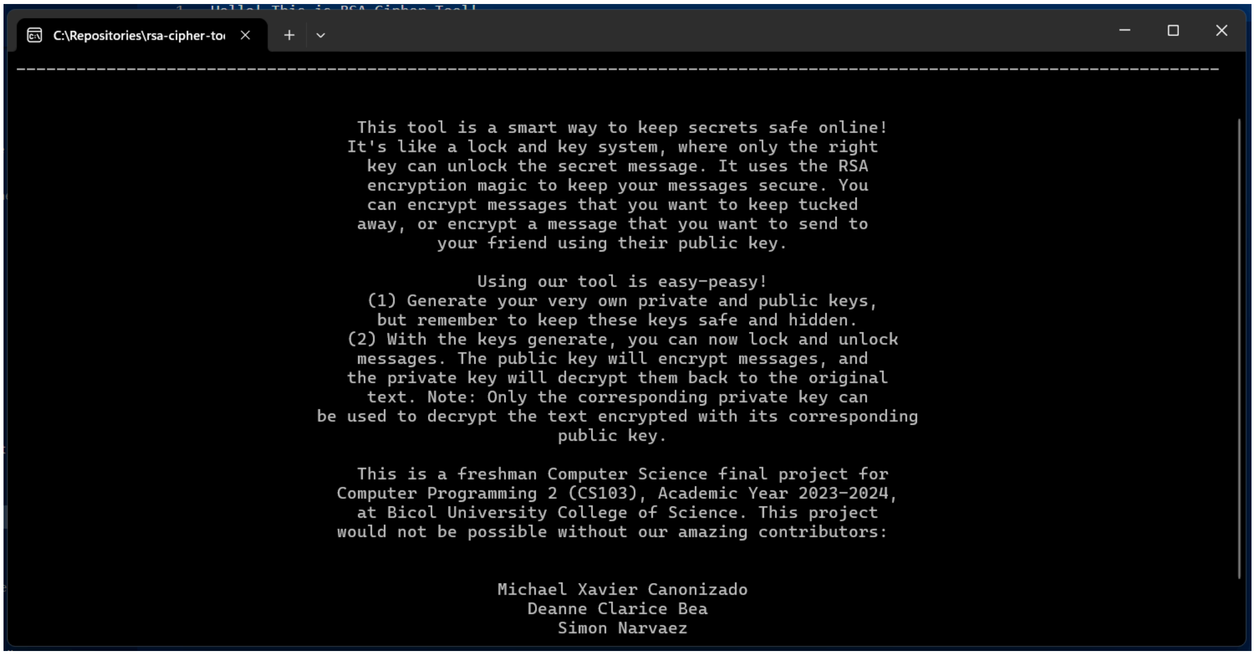
```
------------------------------------------------------------------------------------------------------------------------
                                                     RSA Cipher Tool
------------------------------------------------------------------------------------------------------------------------

Action: Decryption
File name: en.txt
Private key: 752689199.3416175289
Decryption progress: [==========================================================================================] ( 100% )
Decryption complete!
Decrypted file in: 123.54 seconds
Characters decrypted: 31
View the decrypted file at: dc.txt









                                                 Enter DONE to go back: 
```

4) About

This section describes the functionalities of the program and how it can be useful in securing messages that will be sent to other people. It also details the contributors of the program and the reason behind the it.



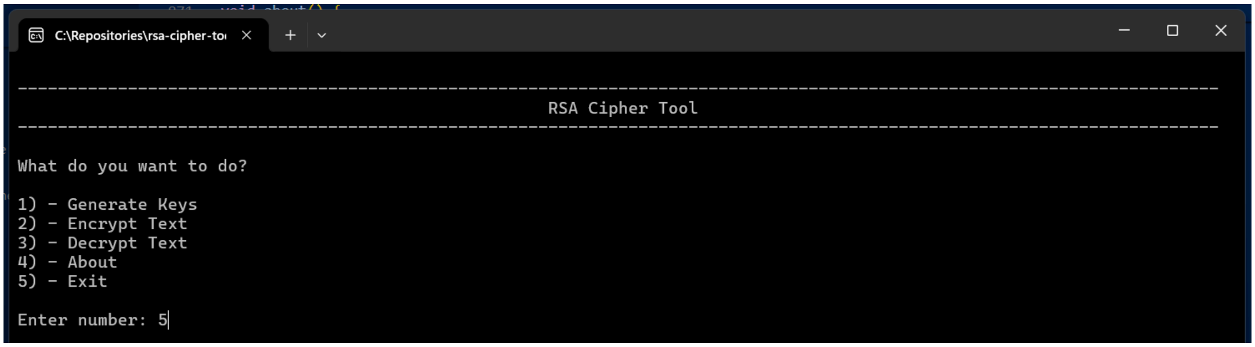
```
------------------------------------------------------------------------------------------------------------------------
                                                     RSA Cipher Tool
------------------------------------------------------------------------------------------------------------------------

What do you want to do?

1) - Generate Keys
2) - Encrypt Text
3) - Decrypt Text
4) - About
5) - Exit

Enter number: 4
```



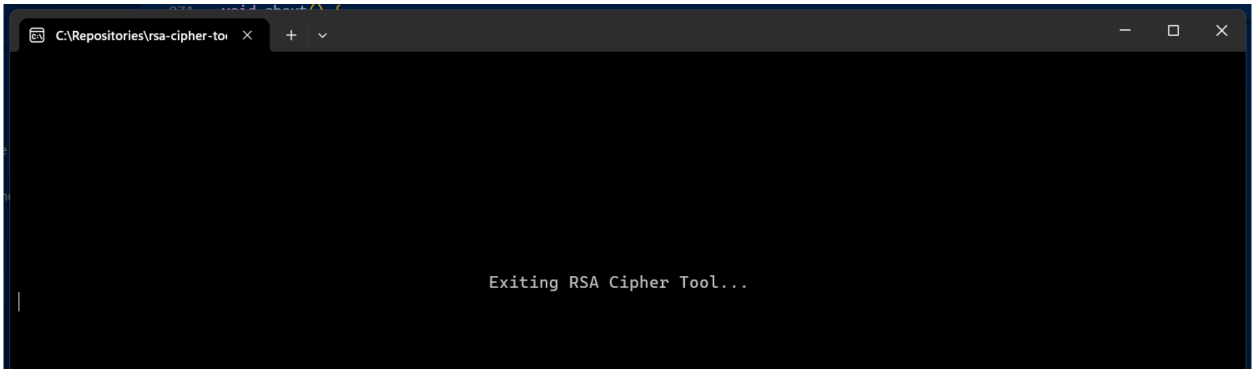
```
------------------------------------------------------------------------------------------------------------------------


                                   This tool is a smart way to keep secrets safe online!
                                  It's like a lock and key system, where only the right
                                    key can unlock the secret message. It uses the RSA
                                    encryption magic to keep your messages secure. You
                                    can encrypt messages that you want to keep tucked
                                   away, or encrypt a message that you want to send to
                                           your friend using their public key.

                                               Using our tool is easy-peasy!
                                    (1) Generate your very own private and public keys,
                                     but remember to keep these keys safe and hidden.
                                  (2) With the keys generate, you can now lock and unlock
                                   messages. The public key will encrypt messages, and
                                  the private key will decrypt them back to the original
                                    text. Note: Only the corresponding private key can
                               be used to decrypt the text encrypted with its corresponding
                                                        public key.

                                   This is a freshman Computer Science final project for
                                 Computer Programming 2 (CS103), Academic Year 2023-2024,
                                   at Bicol University College of Science. This project
                                 would not be possible without our amazing contributors:


                                                Michael Xavier Canonizado
                                                   Deanne Clarice Bea
                                                      Simon Narvaez
```

5) Exit

This option allows the user to exit the program properly with all the memory used deallocated.



```
--------------------------------------------------------------------------------------------------------------------------
                                                      RSA Cipher Tool
--------------------------------------------------------------------------------------------------------------------------

What do you want to do?

1) - Generate Keys
2) - Encrypt Text
3) - Decrypt Text
4) - About
5) - Exit

Enter number: 5
```



```
                                              Exiting RSA Cipher Tool...
```

I.1.E) Member Participation

List here the roles and detailed contributions made by each member.

GITHUB LINK: https://github.com/michaelcanonizado/rsa-cipher-tool

CANONIZADO, Michael Xavier:

- Programmed bignum.h and Bignum.c
- Managed commits of group members
- Implemented the RSA encryption algorithm using bignum.h in rsa-cipher-tool.c
- Documented bignum.h, bignum.c, and rsa-cipher-tool.c
- Refactored and finalized bignum.h, bignum.c, and rsa-cipher-tool.c

BEA, Deanne Clarice:

- Made initial contributions in the rsa-cipher-tool.c until it was working properly with documentations. This was later refactored by Michael Xavier Canonizado with significant contributions. The following are the user-defined functions created in the initial code:

  These functions have conditional compilation as the program is expected to run in both Windows and Linux systems, and thus, required to have distinctive function definitions to work.
  - getTerminalSize
  - clearScreen

- ○ sleepProgram
- ○ moverCursor
- ○ clearLines
- ○ loadingBar

These functions are made to have cleaner code in the main function. Some of the algorithms are used too often which makes it better to construct them as functions (i.e. waitForDone, displayMessage). Some algorithms are too lengthy to be included in the main function. These algorithms have specific purposes and do not have any relations to other ones. Creating a function enables the code to be more comprehensible

- ○ waitForDone
- ○ getConfirm
- ○ displayMessage
- ○ displayAbout
- ○ getInputFile
- ○ getKeys
- ○ getFileSize
- ○ generate
- ○ generateKeys
- ○ encryptTextFile
- ○ encryptText
- ○ decryptTextFile
- ○ decryptText
- ○ aboutProgram

- Helped in the following areas of the project documentation:
  - ○ Algorithm
    - Provided a concise pseudocode of the main function of the program excluding the getting the cursor position and moving the cursor for displaying outputs
  - ○ Program Specifications
    - ■ Standard C functions used
      - Added predefined functions used in both rsa-cipher-tool.c and bignum.c
    - ■ Functions
      - provided some of the detailed description in terms of its purpose, the parameters used, return values, local variables and operations of the user-defined functions.
  - ○ User Manual

    Provided concise explanation of how the program works and included images of the display when particular options are chosen by the user
- Helped in proofreading bignum.c, bignum.h, and rsa-cipher-tool.c

NARVAEZ, Simon:

- Integrated all needed data for the documentation, ensured that all relevant information was included in an organized manner that is simple and easy to understand. Details covered:

  I. Introduction
     - Made a concise yet comprehensive overview of the project, highlighting its core features and functionalities. This introduction serves to captivate the reader's interest and present the project in its best light.

  II. Algorithm
     - Provided a pseudocode that is a clear and structured outline of the system's functionality and operations that allows users to visualize and test out different approaches in implementing features. Later on revised.

  III. Program Specifications

        - A.Standard Functions used in rsa-cipher-tool.c
            - Have identified the predefined functions used on the program. Compiled data through a table.
        - B. Functions used
            - Have carefully provided some of the detailed description in terms of its purpose, the parameters used, return values, local variables and operations of the user-defined functions.

    IV. User Manual
        - Provided a short overview of the program.

- Made the comprehensive Progress Report, a requisite component of this project. This report provides a detailed account of the project's development stages, milestones achieved, and challenges encountered. Its inclusion enriches the documentation, offering valuable insights into the project's evolution and accomplishments.

CAMPOPOS, Marc Jordan:

-