Name: Muhammad Zaeem Shahzad
Student ID: ms12297
University ID: N16933069
Date: November 14, 2020

Assignment 3 Report: Image Steganography

Step 1 - Problem Identification and Statement:

The program displays hidden messages in a bitmap image of the user's. It also hides (with minimal visibility) a message of the user's choice into a bitmap image.

Step 2 - Gathering Information:

- *Encoding:*

Secret messages are encoded into a bitmap image by hiding them in the image using Least Significant Bit Steganography. Every least significant bit of the image pixels is replaced with the bits of the secret message. This makes the message unrecognizable for the common viewer. At the end of the secret message a delimiter character '\0' should be embedded to pixel bytes so the message can be extracted back in the decode process.

- *Decoding:*

The secret message hidden in a bitmap image using LSB Steganography can be extracted by the process of decoding. Every LSB of the pixel bytes is extracted and every 8 bits constitute a character (1 byte). These characters in succession form the whole secret message.

- *Inputs/Outputs:*

The input will be the name of the bitmap image to be encoded or decoded. In case of decoding, the secret message will be displayed as output. In case of encoding, the secret message will be recorded from the user and hidden in the bitmap image using LSB Steganography. The new pixel bytes are then written into the image and the bitmap image with a secret message is created as output.

- *Main Menu:*

  To choose how the program should proceed after starting, the user will be prompted to input characters. These characters can only be A, B, or C to choose between encoding, decoding, or exit program (options displayed on screen). Input of any other character will *result in an error message being displayed and the user will be directed to the "Return to menu or Exit" choice.*

- *For input of A (User decides to encode a secret message in a bitmap image):*
  - *The inputs are the name of the bitmap image file and the secret message to be encoded*
  - *The output is the integration of the secret message into the image file using LSB Steganography.*

- *For input of B (Decoding a bitmap image)*
  - *The input is the name of the bitmap image file with the hidden secret message.*
  - *The output is the secret message.*

- *For input of C (Exit Program)*
  - *The user is directed to the "Return to Main Menu or Exit" choice.*

- *In addition to the inputs mentioned above, a number input to "return to the main menu" or to "exit the program" will also be prompted. This input can be only one of two integers: 1 or 0. Output "Invalid input!" messages will be displayed if any other number or character is input and the user will be directed back to the main menu (start of the program).*

Step 3 - Test Cases and Algorithm:

1) *Test Cases:*

- *Test Case 1 – Encode option chosen, then Return to Main Menu:*

Upon selection of Encode (input of A) from the Main Menu, the program prompts the user for the name of the bitmap image file to encode in. For the input of "NYUAD.bmp", the program prompts the user for the secret message. After input of "Hello!!!", the program displays a message for the completion of the encoding process: "Message encoded successfully". Then the "Return to Main Menu or Exit" choice is displayed. Upon input of 1, the user is directed back to the Main Menu.

- *Test Case 2 – Decode option chosen:*

The Decode option is selected with the input of B. The program prompts the user for the name of the file with the secret message. Again, "NYUAD.bmp" is the input to extract the message. The output is the hidden message displayed on console which was encoded in the previous test case: "Hello!!!"

- *Test Case 3 – Exit Program option chosen, then termination of program*

Upon input of C in the Main Menu, the program displays the confirmation message, "Please confirm exit". Then the "Return to Main Menu or Exit" choice is displayed. Upon input of 0, the program terminates.

2) *Algorithm:*

```
Main Function:

Pass into function: Nothing

Print "Welcome to the Image Steganography Program"
Leave a line

Declare variables:
Integer x
Character choice
Character triple pointer variable imageData
Integer imageWidth
Integer imageHeight
Array of charactes fileName, with max elements 30
A variable to hold a sequence of characters message
Inter z, initially equal to 1

As long as x=1, repeat:

   Print "Choose how the program should proceed"
   Print "There are three routes: Encode a secret message into a bitmap
   image", leave a line
   Print "Decode a secret message from a bitmap image", leave a line,
   "Or exit program"
   Print "To" and go to the next line
   Print towards the left of the console in 3 columns of width 15, in
   the same line:
   Column 1: Print "Encode:"
   Column 2: Print "Decode:"
   Column 3: Print "Exit Program:"
   Go to next line

   In this line, using the same format, print:
```

```
Column 1: "Enter A"
Column 2: "Enter B"
Column 3: "Enter C"
Go to next line

Prompt for input of choice
Select 1 of 4 cases based on input for choice:
For choice being 'A':
 Print "Enter the name of the bitmap image file you wish to hide
 the message in"
 Prompt for input of fileName

 Call the function ReadBitmapImage with inputs fileName, imageData,
 imageWidth, and imageHeight. If the value returned is not true
 then end the selection case

 Print "Enter the secret message to be hidden:"
 As long as z is equal to 1, repeat the following code:
   Prompt for input of message
   Declare integer sizeMessage and assign to it the value of the
   length of message multiplied by 8
   Declare a constant integer sizeImage and assign to it the value
   of imageHeight multiplied by ImageWidth multiplied by 3
   If sizeMessage is greater than or equal to sizeImage:
      Print "The message you entered is too big to be hidden into
      the image file"
      Print "Please enter a shorter message"
   Assign 0 to z

 Assign 1 to z

 Call the encodeImage function with inputs imageData, imageWidth,
 imageHeight, and message.
 Print "Message encoded successfully"

 Call the function WriteBitmapImage with inputs fileName,
 imageData, imageWidth, and imageHeight. If the value returned is
 not true then end the selection case

 Call the function ReleaseMemory with inputs imageData,
 imageHeight, and imageWidth
 End selection process for case 'A'

For choice being 'B':
 Print "Enter the name of the bitmap image file containing the
 hidden message"
 Prompt for input of fileName
 Print "Secret message: "

 Call the function ReadBitmapImage with inputs fileName, imageData,
 imageWidth, and imageHeight. If the value returned is not true
 then end the selection case
```

```
    Call the decodeImage function with inputs imageData, imageWidth,
    imageHeight
    Leave a line

    Call the function ReleaseMemory with inputs imageData,
    imageHeight, and imageWidth
    End selection process for case 'B'

  For choice being anything other than A, B or C:
  Print "Error! Invalid Selection", then go to the next line

  Print "To"
  Print all the following outputs to the left of the screen
  Print in 3 columns, each of 35 units in the same line:
  "Return to the main menu:" and "Exit the program:"
  In the second line, use the same format to print:
  "Enter 1" and "Enter 0"

  Prompt for input of integer variable x
  If x is not a number then:
  Clear input buffer memory for x
  Print "Invalid selection! You will now be returned to the main menu"
  Assign 1 to x

  If x is not 1 and x is also not 0:
  Print "Invalid selection! You will now be returned to the main menu"
  Assign 1 to x

Endpoint of main loop, the loop may repeat or end which will also end
the program based on the input for x.

Pass out: integer 0

End of Main Function

Sub-programs/Function Definitions:

For ReleaseMemory function:

Pass into function: Character triple pointer variable imageData,
integer imageHeight, and integer imageWidth

As long as integer row, initially 0, is less than imageHeight repeat:
  As long as integer col, initially 0, is less than imageWidth repeat:
    Delete the memory dynamically allocated to imageData at the row
    element row and column element col
    Add 1 to col

  Delete the memory dynamically allocated to imageData row row
  Add 1 to row
```

```
Delete the memory dynamically allocated to imageData

Pass out of function: Nothing
End of ReleaseMemory function

For encodeImage function:
Pass into function: Character triple pointer variable imageData,
integer imageWidth, integer imageHeight, and a variable message that
holds a sequence of characters

Declare integer size and assign the length of message to it
Declare a character pointer keys and assign the address of the first
character in message to it
Declare integer BIT, initially equal to 0
Declare integer counter, initially equal to 0

As long as integer row, initially 0, is less than imageHeight, repeat:
   As long as integer col, initially 0, is less than imageWidth,
   repeat:
     As long as integer j, initially 0, is less than 3, repeat:
       Declare character a and assign the element of imageData at row
       index row, column index col, and color index j to it
       Declare character b and assign the element of keys at index
       counter to it
       Declare integer LSB and assign the bit value of b at the BIT
       number bit to LSB
       If the least significant bit of a is not equal to LSB:
         Set least significant bit of a as 0, then if either LSB or
         least significant bit are 1, set least significant bit of a as
         1 and assign the new binary value of a to a
         Assign a to the element of imageData at row index row, column
         index col, and color index j
       Add 1 to BIT
       If BIT is equal to 8:
         Add 1 to counter
         Assign 0 to BIT
       If counter is equal to size plus 1:
         Assign imageHeight to row
         Assign imageWidth to col
         Assign 4 to j
         Break repetition cycle
       Add 1 to j
     Add 1 to col
   Add 1 to row

Pass out of function: Nothing
End of encodeImage function


For decodeImage function:
```

```
Pass into function: Character triple pointer variable imageData,
integer imageWidth, and integer imageHeight

Declare an integer array LSBarray with size 8
Declare an integer LSBsize, initially equal to 0
As long as integer row, initially 0, is less than imageHeight, repeat:
   As long as integer col, initially 0, is less than imageWidth,
   repeat:
     As long as integer j, initially 0, is less than 3, repeat:
       Assign the least significant bit of imageData at row index row,
       column index col, and color index j to LSBarray element at index
       LSBsize

       If LSBsize is equal to 8:
         Assign 0 to LSBsize
         Declare character n, initially the character 0
         As long as integer i, initially 0, is less than 8, repeat:
           If LSBarray element at index i is equal to 1:
             Set the i-th bit of n to 1
           If LSBarray element at index i is equal to 0:
             Set the i-th bit of n to 0
         If n is the character '\0':
           Assign imageHeight to row
           Assign imageWidth to col
           Assign 4 to j
           Break cycle of repetitions

         Print n
         Add 1 to j
     Add 1 to col
   Add 1 to row

Pass out of function: Nothing
End of decodeImage function
```

Step 4 - Code or implementation:

```
/*-------------------------------------------------------------------------------
*/
/* Name: Muhammad Zaeem Shahzad, Student ID: ms12297 */
/* Date: November 14, 2020 */
/* Program: assignment3.cpp */
/* Description: Image Steganography. The program decodes and displays hidden messages in
a bitmap image. It also encodes hidden messages into a bitmap image of the user's
choice*/
/*-------------------------------------------------------------------------------
*/
```

```cpp
#include <iostream> //For input/output
#include <iomanip> //For output manipulation eg; in tabular format
#include <fstream> //Used in BitmapHelper.h, for file-handling
#include <bitset> //For bit-manipulation
#include "BitmapHelper.h"
#include "Decoder.h"
#include "Encoder.h"

using namespace std;

int main() {

	cout << "Welcome to the Image Steganography Program\n";

	int x = 1; //Counter for main loop
	char choice; //For selections in the switch statement
	unsigned char*** imageData; //Pixel array
	int imageWidth;
	int imageHeight;
	char fileName[30]; //To store the name of the bitmap image file
	string message; //The secret message input
	int z = 1; //Counter for message validation

	while (x == 1) {

		//MAIN MENU - Prompting the user to choose how the program proceeds
		cout << "\nChoose how the program should proceed\n" << endl;
		cout << "There are three routes:\nEncode a secret message into a bitmap 
image\n";
		cout << "Decode a secret message from a bitmap image\nOr exit program\n";
		cout << "To\n" << endl;
		cout << left << setw(15) << "Encode:" << setw(15) << "Decode:" << setw(15) 
<< "Exit Program:" << endl;
		cout << left << setw(15) << "Enter A" << setw(15) << "Enter B" << setw(15) 
<< "Enter C\n" << endl;
		cin >> choice;

		switch (choice) { //For execution of commands based on the user's selection

		case 'A':

			cout << "Enter the name of the bitmap image file you wish to hide 
the message in\n";
			cin >> fileName;

			if (!(ReadBitmapImage(fileName, imageData, imageWidth, 
imageHeight))) { //To read the bitmap image file
				break; //If the user inputs an invalid file name, go to Exit 
Display
			}

			cout << "Enter the secret message to be hidden:\n";

			//Validating length of message
			while (z == 1) {
				cin >> message;
```

```
                                long sizeMessage = message.length() * 8;
                                long long sizeImage = imageHeight * imageWidth * 3;
                                if (sizeMessage >= sizeImage) {
                                        cout << "The message you entered is too big to be
hidden into the image file\n";
                                        cout << "Please enter a shorter message: \n";
                                        continue;
                                }
                                z = 0;
                        }
                        z = 1; //Re-initializing z


                        encodeImage(imageData, imageWidth, imageHeight, message);
                        cout << "Message encoded succesfully\n";

                        if (!(WriteBitmapImage(fileName, imageData, imageWidth,
imageHeight))) {//To write the altered pixel bytes into the image file
                                break; //Go to Exit Display
                        }

                        ReleaseMemory(imageData, imageHeight, imageWidth); //Releasing
dynamically allocated memory

                        break;

                case 'B':

                        cout << "Enter the name of the bitmap image file containing the
hidden message\n";
                        cin >> fileName;
                        cout << "Secret message: " << endl;

                        if (!(ReadBitmapImage(fileName, imageData, imageWidth,
imageHeight))) { //To read the bitmap image file
                                break; //If the user inputs an invalid file name, go to Exit
Display
                        }

                        decodeImage(imageData, imageWidth, imageHeight); //Calling the
decodeImage function
                        cout << endl;

                        ReleaseMemory(imageData, imageHeight, imageWidth); //Releasing
dynamically allocated memory
                        break;

                case 'C':
                        cout << "Please confirm your choice\n";
                        break;

                default:
                        cout << "Error! Invalid  Selection" << endl;

                }

                //Exit display
                cout << "\nTo " << endl;
```

```
		cout << left << setw(35) << "Return to the main menu:" << setw(35) << "Exit 
the program:" << endl;
		cout << left << setw(35) << "Enter 1" << setw(35) << "Enter 0\n" << endl;

		//Validating counter for the main menu loop
		//To ensure that x can only be 1 or 0

		if (!(cin >> x)) { //To prompt for input of x AND validate an integer input

			//To clear the buffer memory
			cin.clear();
			cin.ignore(numeric_limits<streamsize>::max(), '\n');

			cout << "\nInvalid selection!\nYou will now be returned to the main 
menu\n" << endl;
			x = 1;
		}

		if (x != 1 && x != 0) { //To validate input of only 1 or 0
			cout << "\nInvalid selection!\nYou will now be returned to the main 
menu\n" << endl;
			x = 1;
		}

	}

	return 0;
}

void ReleaseMemory(unsigned char*** imageData, int imageHeight, int imageWidth) {

	for (int row = 0; row < imageHeight; row++)
	{
		for (int col = 0; col < imageWidth; col++)
		{
			delete[] imageData[row][col];
		}

		delete[] imageData[row];
	}

	delete[] imageData;

}

void encodeImage(unsigned char*** imageData, int imageWidth, int imageHeight, string 
message) {

	int size = message.length();

	//Copying contents of message to a char array
	char* keys = &message[0];

	int BIT = 0;
	int counter = 0; //To compare with size of message

	for (int row = 0; row < imageHeight; row++)
	{
		for (int col = 0; col < imageWidth; col++)
		{
			for (int j = 0; j < 3; j++) {

				char a = imageData[row][col][j];
				char b = keys[counter];
				int LSB = ((b >> BIT) & 1); //Get the bit value of b at the 
BIT-th bit

				if ((a & 1) != LSB) { //If LSB of a is not already equal to 
int LSB

					a = a & 0xFE | LSB; //Set LSB of a as int LSB
					imageData[row][col][j] = a;
				}

				BIT++;

				if (BIT == 8) { //Completion of one character
					counter++;
					BIT = 0;
				}

				if (counter == (size + 1)) { //End encoding
					row = imageHeight;
					col = imageWidth;
					j = 4;
					break;
				}

			}
		}
	}

}

void decodeImage(unsigned char*** imageData, int imageWidth, int imageHeight) {

	int LSBarray[8]; //Array to hold the LSB of pixel bytes (dynamically allocated 
memory)
	int LSBsize = 0;

	for (int row = 0; row < imageHeight; row++)
	{
		for (int col = 0; col < imageWidth; col++)
		{
			for (int j = 0; j < 3; j++) {

				LSBarray[LSBsize] = (imageData[row][col][j] & 1);
				LSBsize++;

				if (LSBsize == 8) {

					LSBsize = 0;
					char n = '0';
					for (int i = 0; i < 8; i++) {
						if (LSBarray[i] == 1) {
							n |= 1UL << i; //Set i-th bit to 1
```

```
                                                }
                                                if (LSBarray[i] == 0) {
                                                        n &= ~(1UL << i); //Set i-th bit to 0
                                                }
                                        }

                                        if (n == '\0') {
                                                row = imageHeight;
                                                col = imageWidth;
                                                j = 4;
                                                break;
                                        }

                                        cout << n;
                                }
                        }
                }
        }
}
```

Step 5 - Test and Verification:

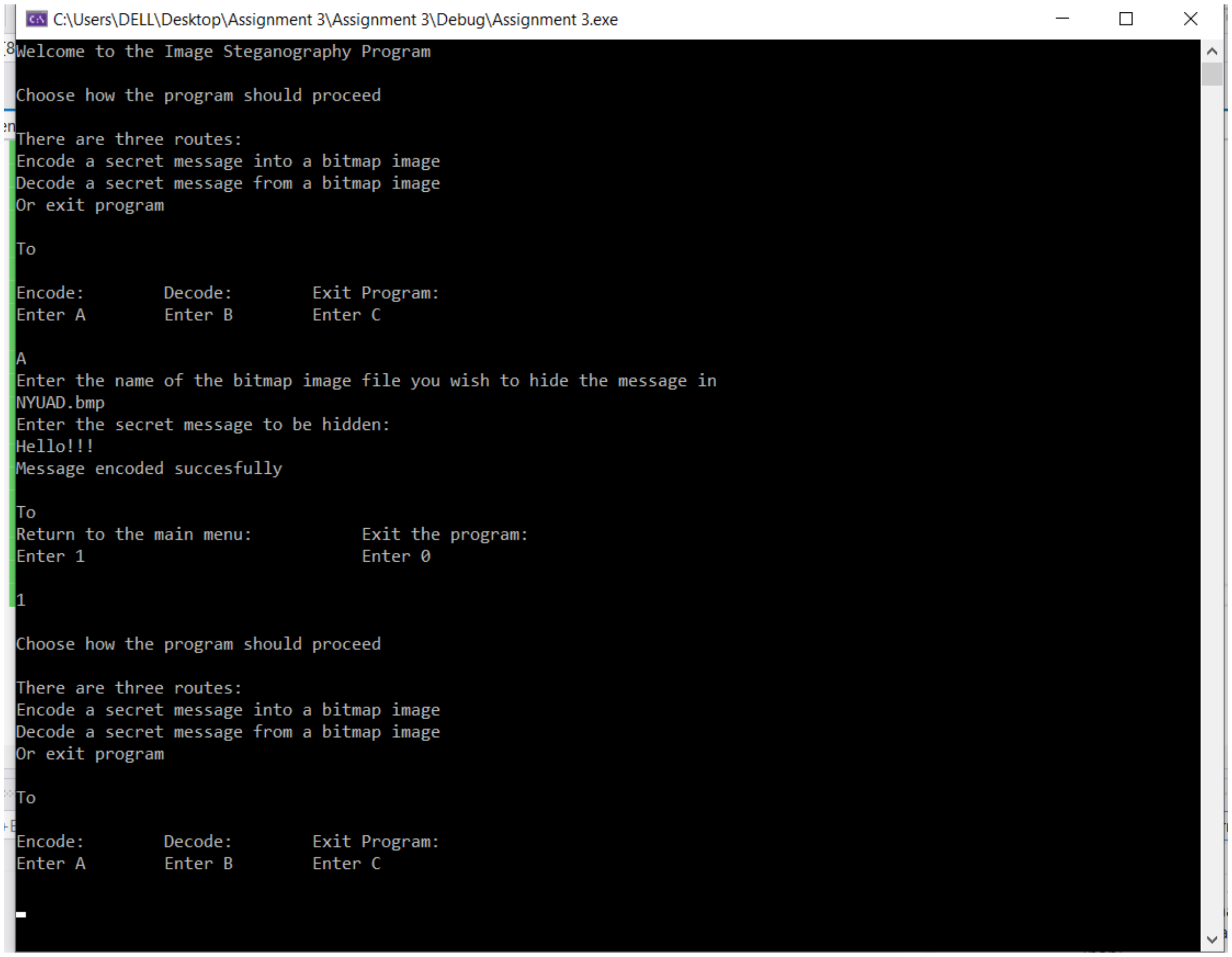
- *Test Case 1 – Print option chosen, then Return to Main Menu:*

  - *For Inputs:*
    - *A*
    - *NYUAD.bmp*
    - *Hello!!!*
    - *1*

  - *Data entered into the program:*
    - *Encode chosen*
    - *Bitmap image file name entered*
    - *Secret message entered*
    - *Return to main menu option chosen*

  - *The Outputs were:*
    - *"Message encoded successfully"*
    - *Main Menu Display*

Console Window for the result of test case 1:


C:\Users\DELL\Desktop\Assignment 3\Assignment 3\Debug\Assignment 3.exe

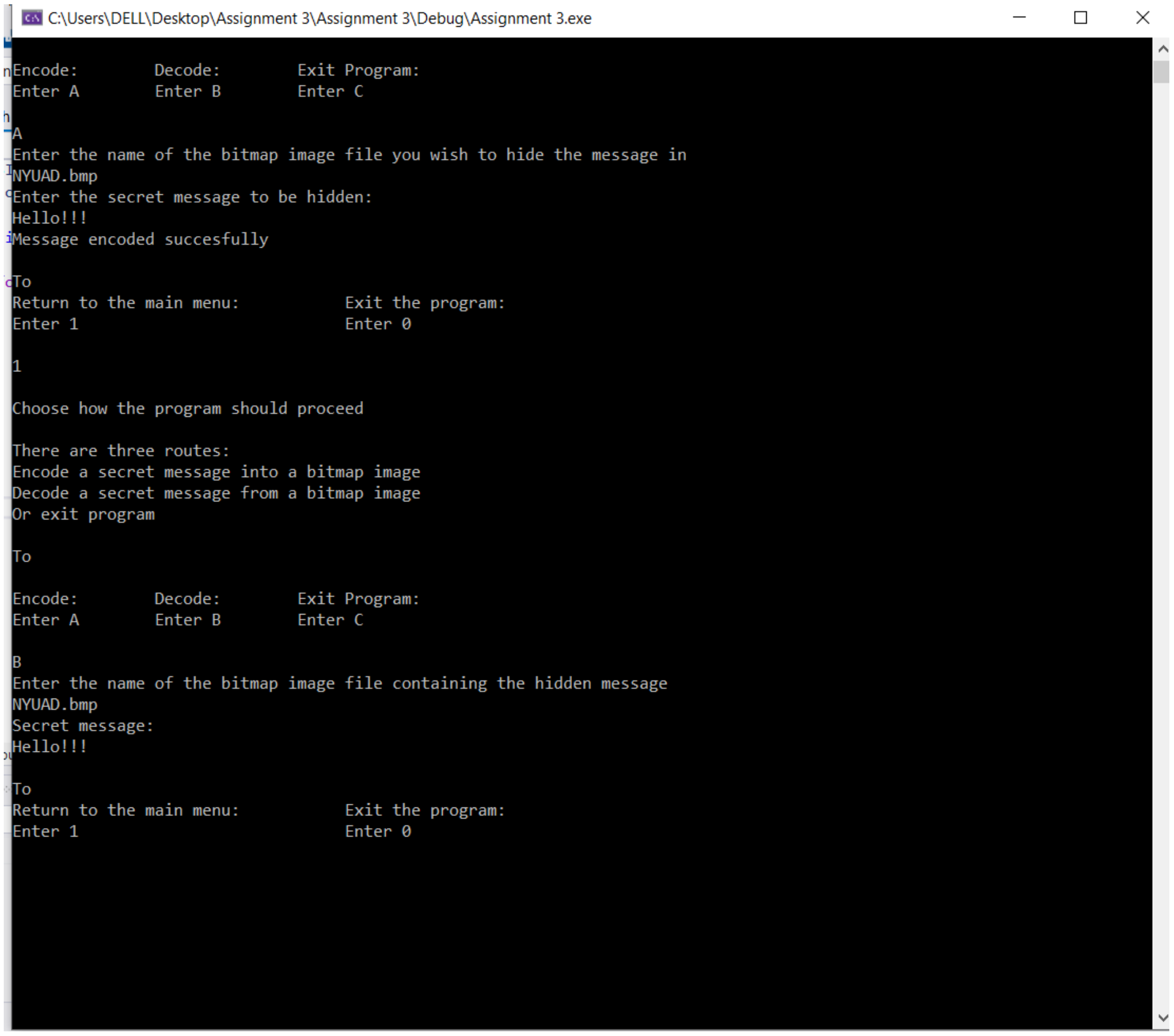
```
Welcome to the Image Steganography Program

Choose how the program should proceed

There are three routes:
Encode a secret message into a bitmap image
Decode a secret message from a bitmap image
Or exit program

To

Encode:        Decode:        Exit Program:
Enter A        Enter B        Enter C

A
Enter the name of the bitmap image file you wish to hide the message in
NYUAD.bmp
Enter the secret message to be hidden:
Hello!!!
Message encoded succesfully

To
Return to the main menu:           Exit the program:
Enter 1                            Enter 0

1

Choose how the program should proceed

There are three routes:
Encode a secret message into a bitmap image
Decode a secret message from a bitmap image
Or exit program

To

Encode:        Decode:        Exit Program:
Enter A        Enter B        Enter C
```

- *Test Case 2 – Decode option chosen*
    - ➢ *For Inputs:*
        - *B*
        - *NYUAD.bmp*

    - ➢ *Data entered into the program:*
        - *Decode chosen*
        - *Bitmap image file name entered*

    - ➢ *The Outputs were:*
        - *"Hello!!!"*
        - *Return to Main Menu or Exit Display*

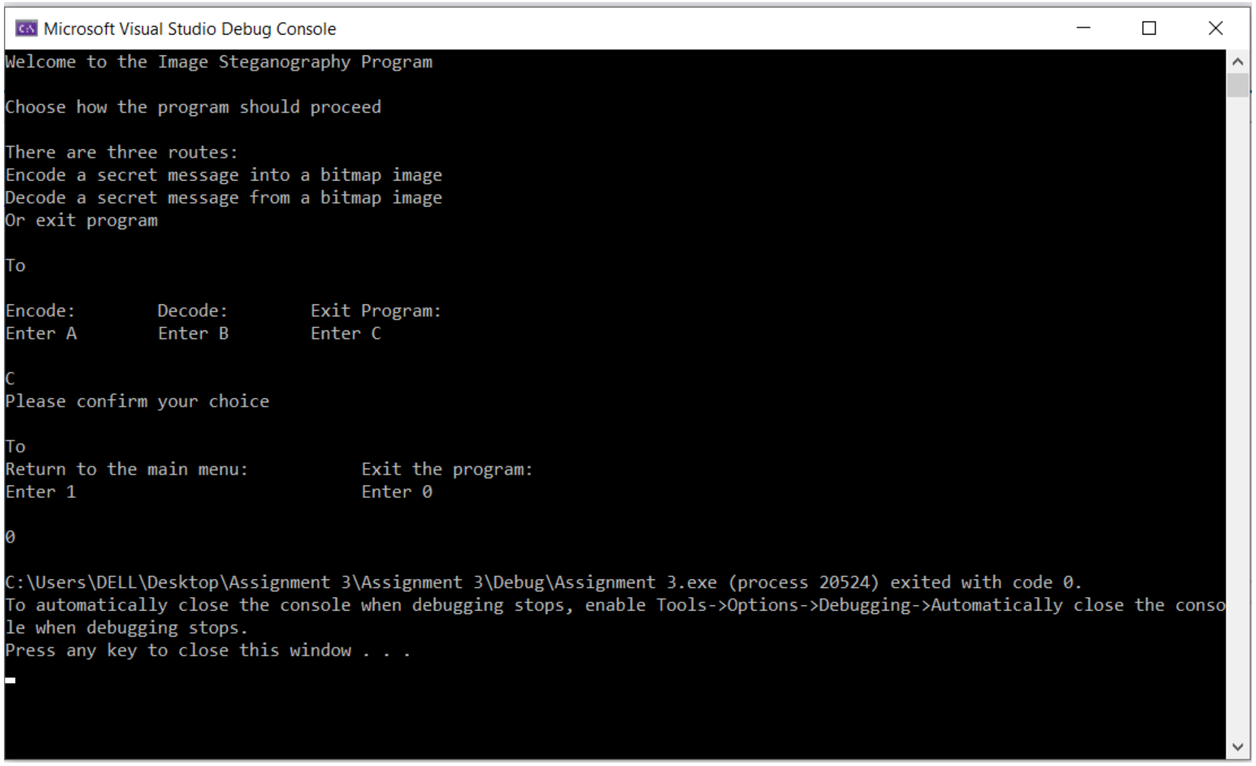
Console Window for the result of test case 2:


C:\Users\DELL\Desktop\Assignment 3\Assignment 3\Debug\Assignment 3.exe

```
Encode:       Decode:       Exit Program:
Enter A       Enter B       Enter C

A
Enter the name of the bitmap image file you wish to hide the message in
NYUAD.bmp
Enter the secret message to be hidden:
Hello!!!
Message encoded succesfully

To
Return to the main menu:          Exit the program:
Enter 1                           Enter 0

1

Choose how the program should proceed

There are three routes:
Encode a secret message into a bitmap image
Decode a secret message from a bitmap image
Or exit program

To

Encode:       Decode:       Exit Program:
Enter A       Enter B       Enter C

B
Enter the name of the bitmap image file containing the hidden message
NYUAD.bmp
Secret message:
Hello!!!

To
Return to the main menu:          Exit the program:
Enter 1                           Enter 0
```

- *Test Case 3 – Exit Program option chosen, then termination of program*

For the inputs of C, and then 0, the program terminates as was expected.
The console window is shown below:



Microsoft Visual Studio Debug Console

```
Welcome to the Image Steganography Program

Choose how the program should proceed

There are three routes:
Encode a secret message into a bitmap image
Decode a secret message from a bitmap image
Or exit program

To

Encode:        Decode:        Exit Program:
Enter A        Enter B        Enter C

C
Please confirm your choice

To
Return to the main menu:           Exit the program:
Enter 1                            Enter 0

0

C:\Users\DELL\Desktop\Assignment 3\Assignment 3\Debug\Assignment 3.exe (process 20524) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
le when debugging stops.
Press any key to close this window . . .
```