

## Step 1: Problem Identification and Statement

The objective of our program is stated in the document Assignment+3+--+Image+Steganography.pdf as follows:

“The software presents the user with a main **menu**, with (three) options to **encode a message**, **decode a message**, and **exit** the program”.

In our context, encoding a message means to hide a secret text message inside a Bitmap image. Both secret message and name of Bitmap image are provided by the user.

Decoding means to extract a secret message embedded in the encoded bitmap image. The name of the encoded bitmap image is provided by the user.

Solution must be arranged into separate header files (encoding.h and decoding.h ), and program must be written in reusable functions.

Bitmap images will be read and written on through functions provided in header file bitmapHelper1.h.

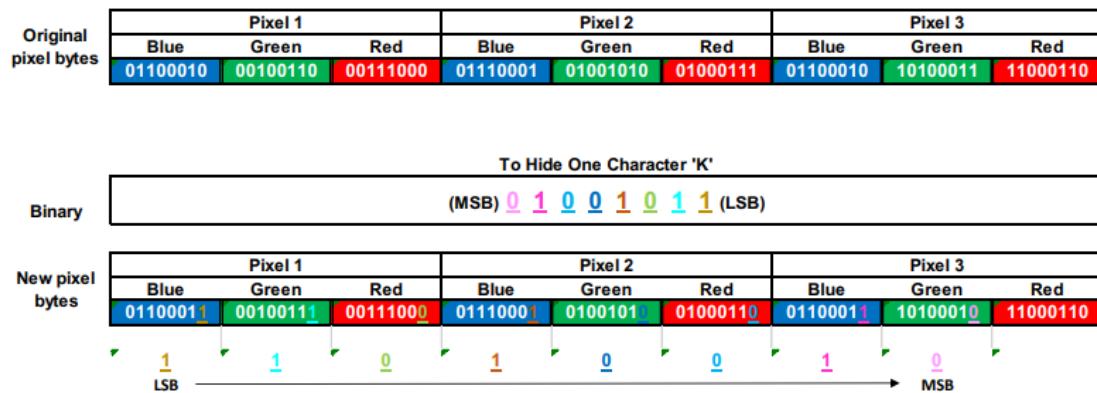
Release memory function should be added to bitmapHelper1.h to deallocate dynamic memory correctly

## Step 2: Gathering Information

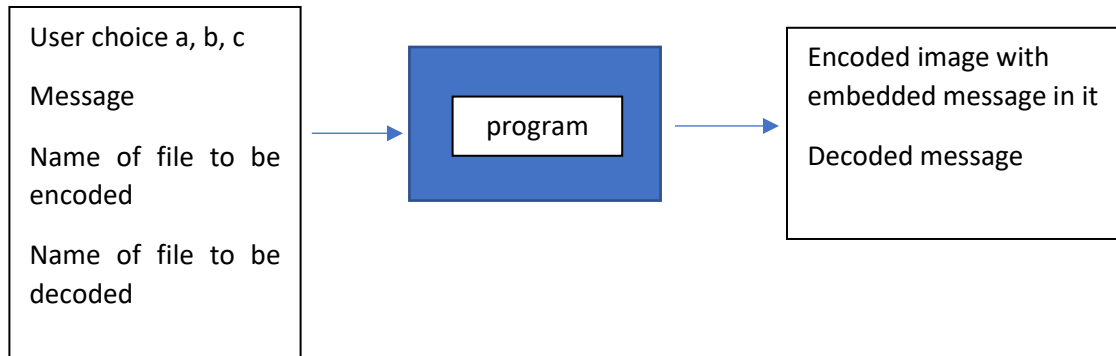
The software makes use of the technique of Steganography which hides “ data (file, text, image, etc.) within another data file (file, text, image, etc.)” . Which can be implemented as follows:

- To encode: find ascii of each letter in message provided by casting character variable to integer
- Convert the ascii of each letter to its binary form by creating a function which converts decimal form of a number to its binary
- Take the least significant bit (lsb) of the binary of each pixel channel through bitwise operators. (Eg. `lsbpixel = (imageData[i][j][channel] >> 0) & 1U`)
- Compare this lsb with the most significant bit of character provided in its binary form
- If they differ toggle (change from 0 to 1 or vice versa) the last bit in the binary of each pixels channels through bitwise operators. (Eg. `imageData[i][j][channel] ^= 1UL << 0`)
- To decode: store lsb of the binary of each pixel channel by groups of 8 bits (1 character)
- Find the ascii corresponding to the character by creating function which converts binary form of a number to decimal
- Cast the character's ascii to type character
- Display character.

This procedure is more successfully explained by professor through the following image:



The **Input/output** diagram of our program is provided as follows:



### Step 3: Test Cases and algorithm

#### Test Cases

The main part of our program which will be tested are: the menu, encoding, decoding and release memory.

#### Menu

input	Expected output
a	"encoding selected"
b	"decoding selected"
c	"exit was selected"
3	Program exits

#### Encoding

Input: secret message:" Mario won"; name of two bmp images to be encoded "Earth.bmp" "earthhh" "Earth" and "NYUAD.bmp".

Two of the file names are invalid to check whether our program correctly breaks. Two images are tested to check whether our program can encode any bitmap image. Our message contains a space to check whether the program can deal with characters other than letters and display them correctly.

Expected output for this section is 0, signaling that code was executed correctly

## Decoding

Input: encoded images with encoded message "Mario won"

Expected Output: Sentence "Mario won" on output screen. A space must be present between the words Mario and won. No other characters should appear after won.

## Release Memory

If program encodes and decodes correctly implies memory in heap is correctly deleted, otherwise program would crash. Hence, release memory function won't be tested.

## Algorithm

### Menu

Define character variable choice;

Print "Enter 'a' to encode a message, 'b' to decode a message, 'c' to exit program" ;

Assign input character to choice;

If choice equals 'a':  
Print "encoding selected";  
Call function "encode";  
Exit loop;

Otherwise, if choice equals 'b':  
Print "decoding selected";  
Call function decoding;  
Exit loop;

Otherwise, if choice equals 'c':  
Print "exit was selected";  
Exit loop;

Output '0' to user.

## Encoding.h

### DecimalToBinary

Define function DecimalToBinary which will not return values;  
DecimalToBinary requires an integer number n and an array of integers as input;

For index of array i starting from the greatest, until integer number is greater than 0, and decrementing the index i of array for each iteration:

Assign to element of array with index i the modulus of n divided by 2;  
Assign n/2 to n;

For index of array j starting from 0, until the index equals bitperbyte and incrementing j at each iteration:  
    if the element of array with index j does not equal 1  
    then assign 0 to that element of array with index j;

### RevertBinary

Define function RevertBinary which will not return values;  
RevertBinary requires two arrays of integers values;

For index of array a starting from 0, until the index a equals bitperbyte and incrementing a at each iteration:

    assign to element of array with index a, the element of that same array with index (bitperbyte-1) -a

### encode

assign 8 to constant integer variable bitperbyte;  
assign 50 to constant integer variable messagelength;  
assign 3 to constant integer variable channelsperpixel;

Define function encode which will not return values;  
encode does not require input arguments;

assign 0 to imageHeight;  
assign 0 to imageWidth;  
define pointer to pointer to pointer of characters imageData;

define string namefile;  
print "Please enter name of file to decode"  
assign input to namefile;

call ReadBitmapImage, by converting namefile into pointer to char array which is input,  
take imageHeight, imageWidth, imageData as input;

if output value of ReadBitmapImage is 1  
    the print "file opening successful";  
otherwise  
    exit program returning -1;

ignore space as input;  
define string secret message;  
define variable pointing to character secretmessagearray;

Print "please enter the secret message to be encoded ";  
Assign line provided as input to secretmessage  
Add delimiting character to secretmessage

Assign array of character from string secret message to secretmessage array

```

secretmessagearray = secretmessage.c_str(); //secret message string is converted in
array of characters

define 1d array of integers messagetable of size [messagelength * bitperbyte];
define pointer to integer messagetablePtr;

for (counter i=0, secretmessagearray[i] is not '\0', increment i at each iteration:)
    cast to integer secretmessagearray[i] and assign it to asciicharacter;
    asciicharacter = (int)secretmessagearray[i]; //convert
    define 1d array of integers named binary of size bitperbyte
    call function DecimalToBinary using as inputs asciicharacter, and binary;
    define 1d array of integers revertedbinary of size bitperbyte;
    call function RevertBinary using as inputs binary and revertedbinary;
    for counter j=0, until j=bitperbyte, incrementing j at each iteration
    for (int j = 0; j < bitperbyte; j++) {
        assign revertedbinary[j] to value messagetablePtr is pointing to
        increment address in messagetablePtr

assign address messagetable to messagetablePtr;

For counter i=0, until i= imageHeight, incrementing i for each iteration,
    For counter j =0, until j= imageWidth, incrementing j at each iteration,
        For channel = 0; until channel = channelsperpointer, incrementing channel
        at each iteration,

            Assign last digit of imageData[i][j][channel] to lsbpixel;
            If lsbpixel does not equal value pointed by messagetablePtr
            if (lsbpixel != *messagetablePtr)
                toggle last digit of imageData[i][j][channel],
            if (value pointed by messagetablePtr does not equal 0 and it
does not equal 1)
                exit the loop;
            increment address stored in messagetablePtr;

call function WriteBitmapImage given as input: namefile.c_str(), imageData, imageHeight,
imageWidth;

call function ReleaseMemory given as input: imageData, imageHeight, imageWidth;

```

## Decoding.h

### decoding

Define function `decoding` which will not return values;  
decoding does not require input arguments;

```

assign 0 to imageHeight;
assign 0 to imageWidth;
define pointer to pointer to pointer of characters imageData;

define string namefile;
print "Please enter name of file to decode"
assign input to namefile;

```

call ReadBitmapImage, by converting namefile into pointer to char array which is input,  
take imageHeight, imageWidth, imageData as input;

```
if output value of ReadBitmapImage is 1
    the print "file opening successful";
otherwise
    exit program returning -1;
```

define array of integers named binary of size bitperbyte;  
assign 0 to n;

define character variable named character;

For counter i=0, until i= imageHeight, incrementing i for each iteration,  
 For counter j =0, until j= imageWidth, incrementing j at each iteration,  
 For channel = 0; until channel = channelsperpointer, incrementing channel  
at each iteration,

```
    Assign last digit of imageData[i][j][channel] to lsbpixel
    Assign lsbpixel to binary[n]
    If n smaller than (bitperbyte-1) {
        Increment n
        Ignore code below and start next iteration,
    otherwise if n equals (bitperbyte-1)
        Assign result of BinaryToDecimal(binary) to ascii
        if ascii equals 0 or ascii smaller than 1
            exit loop;
        if I greater than
            exit loop;
    assign to character, value of ascii casted to character type;
    print character;
    assign 0 to n;
```

call function ReleaseMemory given as input: imageData, imageHeight, imageWidth;

### BinaryToDecimal

Define function BinaryToDecimal which will return a variable of type double;  
BinaryToDecimal requires an array of integers as input;

```
assign 0 to decimal;
assign 0 to factor;
```

For index of array j starting from 0, until the index equals bitperbyte and incrementing  
j at each iteration:

```
    Assign  $2^j$  to factor;
    Assign to decimal: decimal +  $2^j$  x revertedbinary[j] ;
return decimal;
```

BitmapHelper1.h

### ReleaseMemory

Define function ReleaseMemory which will not return values;  
Release memory requires as input: pointer to pointer to pointer, two integers values  
imageHeight and ImageWidth;

```
For counter i=zero, until i= imageHeight, incrementing i for each iteration,  
    For counter j =0, until j= imageWidth, incrementing j at each iteration,  
        Delete dynamically allocated array imageData[i][j],  
        Delete dynamically allocated pointed array imageData[i],  
Delete dynamically allocated array imageData,
```

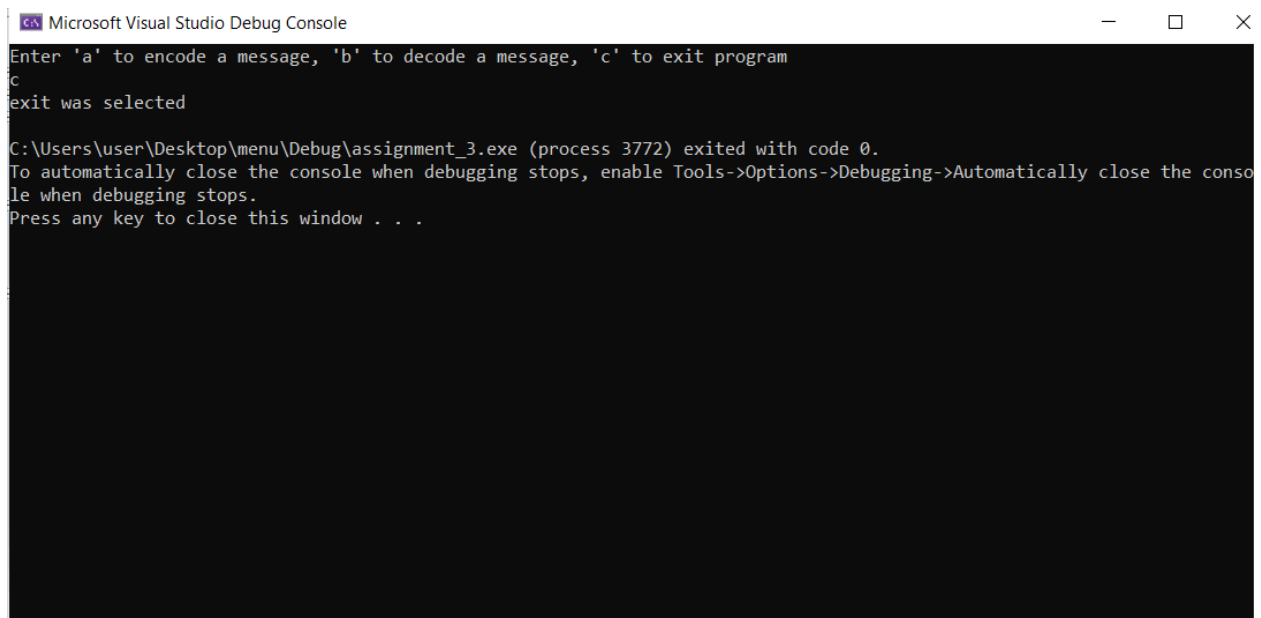
### d) Step 4: Code or implementation

The code is presented in the attached document menu.cpp and header files encoding.h, decoding.h, BitmapHelper1.h

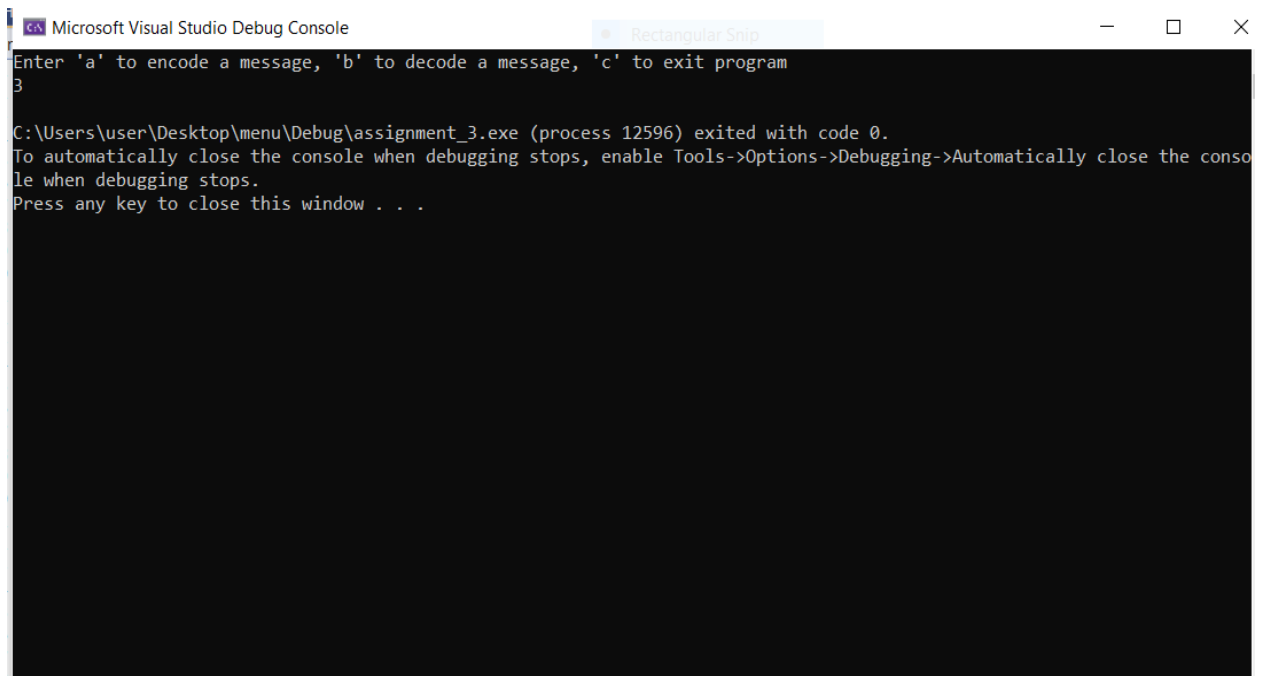
### e) Step 5: Test and Verification

Exit Menu

When c is selected, menu is exited as expected:

A screenshot of the Microsoft Visual Studio Debug Console window. The window has a title bar with the Visual Studio logo and the text "Microsoft Visual Studio Debug Console". The console output shows the following text: "Enter 'a' to encode a message, 'b' to decode a message, 'c' to exit program", followed by a user input "c", then "exit was selected", and finally "C:\Users\user\Desktop\menu\Debug\assignment\_3.exe (process 3772) exited with code 0." Below this, there is a message: "To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops." and "Press any key to close this window . . .". The console window has standard Windows window controls (minimize, maximize, close) in the top right corner.

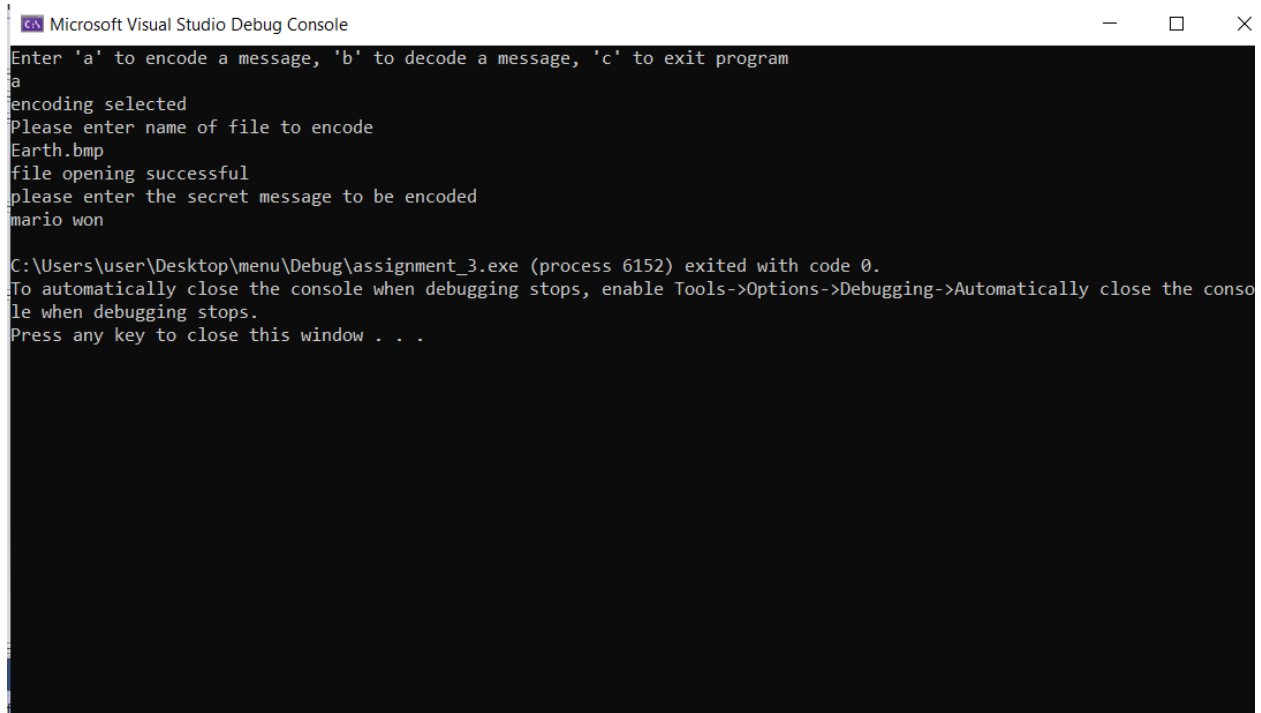
```
Microsoft Visual Studio Debug Console
Enter 'a' to encode a message, 'b' to decode a message, 'c' to exit program
c
exit was selected
C:\Users\user\Desktop\menu\Debug\assignment_3.exe (process 3772) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

A screenshot of the Microsoft Visual Studio Debug Console window. The title bar shows 'Microsoft Visual Studio Debug Console' and a 'Rectangular Snip' watermark. The console output shows the program's instructions: 'Enter 'a' to encode a message, 'b' to decode a message, 'c' to exit program'. The user has entered '3', and the program has exited with code 0. The console also displays a message about automatically closing the console when debugging stops.

```
Microsoft Visual Studio Debug Console
Enter 'a' to encode a message, 'b' to decode a message, 'c' to exit program
3
C:\Users\user\Desktop\menu\Debug\assignment_3.exe (process 12596) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

When input is 3, program exits as expected. We were not asked to correct user, so no corrections are necessary

### Encoding, menu a

A screenshot of the Microsoft Visual Studio Debug Console window showing the program's execution for the 'a' menu option. The user has entered 'a', and the program has printed 'encoding selected'. The user has then entered 'Earth.bmp' as the file name, and the program has printed 'file opening successful'. Finally, the user has entered 'mario won' as the secret message, and the program has printed 'mario won'. The console also displays a message about automatically closing the console when debugging stops.

```
Microsoft Visual Studio Debug Console
Enter 'a' to encode a message, 'b' to decode a message, 'c' to exit program
a
encoding selected
Please enter name of file to encode
Earth.bmp
file opening successful
please enter the secret message to be encoded
mario won
C:\Users\user\Desktop\menu\Debug\assignment_3.exe (process 6152) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

As shown above when a is selected, “encoding selected” message is printed correctly. Which shows the menu is correctly working.



Exit code after encoding is 0 as expected

Following images shows that code breaks when image name or format is not correct as expected

```
Microsoft Visual Studio Debug Console
Enter 'a' to encode a message, 'b' to decode a message, 'c' to exit program
a
encoding selected
Please enter name of file to encode
earthhh
Failed to open the Image file
Failed to open the Image file

C:\Users\user\Desktop\menu\Debug\assignment_3.exe (process 16828) exited with code -1.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

```
Microsoft Visual Studio Debug Console
Enter 'a' to encode a message, 'b' to decode a message, 'c' to exit program
a
encoding selected
Please enter name of file to encode
Earth
Failed to open the Image file
Failed to open the Image file

C:\Users\user\Desktop\menu\Debug\assignment_3.exe (process 16612) exited with code -1.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

Debugging below shows program does encode image NYUAD as well which was expected.

```
Microsoft Visual Studio Debug Console
Enter 'a' to encode a message, 'b' to decode a message, 'c' to exit program
a
encoding selected
Please enter name of file to encode
NYUAD.bmp
file opening successful
please enter the secret message to be encoded
mario won

C:\Users\user\Desktop\menu\Debug\assignment_3.exe (process 3068) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

## Decoding menu b

```
Microsoft Visual Studio Debug Console
Enter 'a' to encode a message, 'b' to decode a message, 'c' to exit program
b
decoding selected
Please enter name of file to decode
Earth.bmp
file opening successful
mario`won

C:\Users\user\Desktop\menu\Debug\assignment_3.exe (process 11172) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

As shown above when b is selected, “decoding selected” message is printed correctly. Which shows the menu is correctly working.

Message mario won is displayed. However, space is substituted by an apostrophe.

```
Microsoft Visual Studio Debug Console
Enter 'a' to encode a message, 'b' to decode a message, 'c' to exit program
b
decoding selected
Please enter name of file to decode
NYUAD.bmp
file opening successful
mario`wonΣm fū%Em' ||æqZ|~ñΓx+U==q~Λf6Vfμ-cfpΣ±v8paL ♡ @α°@αβyÉLGVΣγaG2ÄIé♥δ$U†τl$^+?■▼LQI8Ö8f#2#G||8ππ~π-C||»f~♦LjlηL-J
JR%I||oL|/-||m||m'Nε†Ib||m||φ$φXîÆC&SY9ëY-Nε-qΓ†L ðIraα~n!-LüπLq^n$||+||_||C♥||oL||qLÄπ° δC♥° ▼CJjUUUUU||@C#~Aπ o`lU~Æf6i+
7pLüFRU% C♥ αα°o^n°@nπ8nLθ ü|m;ñQ|U>||o^n°A oαα$Ij+-UN-δ*Uj--R[I²üv o^nC♥C-N-îΣ||æL GnπHrπFnmL Ir||î$α^nα~æ||6r$7rL7Æ||m||qL
fπmÄLc[U%æL'ë!!|φ8|o'ëπ†vE$|c|Ibn||†|ëc;||c;||Y-NL;q~†± C♥C$-ÆTI†±π?ÄπC' j$Uπ†q$IÄq|▼†ü$||îΣ||m||J«-δ*§\v♥q CAð:|÷ü-||ëπ#α≡♥°▼
pCî%πr~>-ê_èö~L[;i9«RñCRñJÆC♥?§fv†||φ-N†;Ib;||m||φñ→fMuqDor$|Y-IÆ8vlt @πqαc@+ñ-Cπ8pa~Hθ.IC♥ αC♥8C♥C @^||π^oαo%|-Z-†π
8Cπ#≡▼C C IÆ†|mσ|ð| ||^n^Loh||&-φθL C♥°@L8C♥#GÄΓc j rΣ=KSC%xn^~Cπ° ▼C|ÆT-δ--RUUjñVm†VR ▼C 8Ä_C||!!ÄΣ|†î$9Ä_||m#||î_Hr_6rΣ8Ä||8^n ||mq#
9Ä_8Ä||m||†qÄ_||mL9r#f8æTi%|-q||IÆ-I_L'Nε-||!!||π$ëYx#9||Äð8||c||I1'v†|IΓ-Äφπ?pLp-ÆT†R+C C♥
C:\Users\user\Desktop\menu\Debug\assignment_3.exe (process 16140) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

Our code is not able to break correctly when decoding image NYUAD.bmp. Which was not expected. Notice that code breaks correctly when decoding image Earth.bmp.

From this analysis we determine that:

The message was encoded and decoded successfully, which is the main objective of our program. Menu is working successfully.

However, the program needs to be improved:

- My program is still not able to recognize delimiter character and stop decoding when delimiter character is reached in the case of NYUAD.bmp. This was unexpected.  
Hence, changes should be done to encoding and decoding.h such that program will be able to detect the limiter character for any bitmap image stop decoding when delimiter character is found.
- Space is not correctly displayed in decoding. Hence, encoding.h should be modified to ensure right binary for character space " " is encrypted in the image. The problem may arise from our decision of using `cin.ignore()` to ignore character in our input message.