**Part 1: Encryption Outline**

My encryption program will work as follows. Each character will go through two layers of conversion, into unicode, and then either into binary or into hexadecimal depending on its index. For each character in the entered message a loop will check if the index of the character is even or odd. If the index is even the character will first be converted into unicode. Then, the unicode will be converted into binary. A different conversion will execute if the index of the character is odd. First, it will again be converted into unicode. Then the unicode will be converted into hexadecimal. After the characters are converted they will be entered into a label with a space between each character to help with decryption. The end result of the encryption is a series of numbers switching from binary to hexadecimal, with a space between each of the letters. The decryption process works by simply reversing the encryption. It identifies the end of each letter using the space. Then if the letter is even it makes a substring of 7 units(since it is binary) and converts it into unicode, and then into a character. If the letter is odd it makes a substring of 2 units and converts into unicode then a character. Each character is then concatenated into a label.

**Part 2: Problems and Solutions**

While working on my encryption program I encountered many different problems. However, I used the resources at my disposal to help me overcome them. One of the problems I experienced is figuring out how to convert a unicode integer to a binary string. I resolved this problem by searching online and finding a function that converts an integer into a string with a specific base. This helped me to convert into binary. Afterwords, I encountered difficulty with the conversion to hexadecimal. With the help of the internet I realized that I could use the same function as binary for hexadecimal. Another problem that I was met with was how to create a substring of each binary string and hexadecimal string, since I didn’t know the length of each value. However, I resolved this by counting the number of characters in each string and realizing that there are always 7 in binary, and 2 in hexadecimal. I also encountered a problem when a letter was missing from the encryption if there was no space entered after the end of the message. I fixed this by finding the Space() function in the Chapter 5 booklet. Beyond these large problems I experienced a few small ones, mainly to do with syntax. I resolved these by asking for help from my classmates.