Key to Cryptography

I finally developed what I think is a fairly elegant solution to this Kattis Problem (it took me longer than it should, I think!)

Here's the code I wrote to solve this problem: Key to Cryptography.

```
1  #include <iostream>
   #include <string>
3  using namespace std;

5  int main()  {
     string cypherText, secretWord;
7    cin >> cypherText >> secretWord;
     string plainText = cypherText;
9    int cypherLength = cypherText.size(), secretLength = secretWord.size();
     int minLength = cypherLength < secretLength ? cypherLength : secretLength;
11   int prt1{}, prt2{};
     for (int i = 0; i < minLength; i++)   {
13     plainText[prt1++] = (char)(((cypherText[i]-secretWord[i]+26) % 26)+65);
     }
15
     for (int i = secretLength; i < cypherLength; i++)   {
17     plainText[prt1++] = (char)(((cypherText[i]-plainText[prt2++]+'A') % 26)+65);
     }
19
     cout << plainText;
21 }
```

Answer the following questions (all essay questions) about this code in the quiz posted on Canvas.

1. Explain why, on line 8, it makes sense to have `plainText` initialized with the value of `cypherText`.
2. Here we'll try to get at what's going on on line 10.
   (a) What is the value of `minLength` if `cypherText = "ULIXYDI"` and `secretWord = "CHIEF"`?
   (b) What is the value of `minLength` if `cypherText = "ULIXY"` and `secretWord = "SEATTLE"`?
   (c) What's the purpose of `minLength` in the for-loop that follows in lines 12-14?
3. Suppose `cypherText = "ULIXYDI"` and `secretWord = "CHIEF"`. Complete the table below to see how the first part of `plainText` is computed in the for-loop of lines 12-14.

threading for-loop				
`prt1`	`i`	`cypherText[i]`	`secretWord[i]`	`(cypherText[i] - secretWord[i] + 26) % 26`
0	0	U	C	18
⋮	⋮	⋮	⋮	⋮

4. Now complete the table below to see how the second part of `plainText` is computed in the for-loop of lines 16-18.

threading for-loop					
`prt1`	`prt2`	`i`	`cypherText[i]`	`plainText[i])`	`(cypherText[i]- plainText[prt2++]+ 'A') % 26`
	0				