NOTE: For this assignment, there are several additional requirements you will be graded on in addition to producing the correct output. They are: (1) you should write an outline of your program design that reflects your top−down functional/modular decomposition. This should be included in your comments at the beginning of your program, (2) your program should contain good documentation, (3) you must write and use at least two functions in your program. The GTAs will be HAND−CHECKING your code on this assignment to grade you on these three items. You should be sure to read and apply the documentation standards as referenced in the Programming Standards section of this specification.

For the outline of your design:
You must include, below the header comments, an outline program design. The design must reflect your top−down functional/modular decomposition of the problem. The design should follow the layout of the payroll design example in appendix 4 of the course notes.

This programming assignment uses many of the ideas presented in sections 5, 6, 7, and 8 of the course notes, so you are advised to read these notes and the following program specification carefully.

The Program Specification: Substitution Cipher

For this assignment you will write a program that implements a substitution encryption algorithm. Your program must do the following:

- Run the decryption/encryption algorithm according to the information in the input file
- Read from the input file `input.dat`
- Write to the output file `output.dat`

Substitution Encryption Algorithm:

Encryption is the process of changing a message in such a such a way that only trusted people can decrypt and read the message. This is achieved by using a "key" to lock the message and giving the key to trusted users, much like a cash box in a bank.

Here are some useful terms pertaining to encryption:

plain text	The message in its original form
cipher text	The message after it is encrypted
key	The information needed to decrypt or encrypt the message

For the purposes of this project, the encryption algorithm is a form of substitution encryption. In substitution encryption every character in the message is substituted with a corresponding cipher character, which is specified by the key. The key is a lookup table of the form:

Plain Character	***Cipher Character***
'A'	'T'
'B'	'G'
'C'	'B'
'D'	'Y'
'E'	'H'
...	...
'X'	'N'
'Y'	'U'
'Z'	'J'
'_'	'M'

The substitutions encryption algorithm works as follows. For each character in the input text, lookup the corresponding cipher character in the lookup table and replace the original character with the cipher character. Continue these operations until all characters in the input text have been replaced. The resulting text is the cipher text.

An Example:

Given the key table listed above and the following input text:

```
BY_ACE
```

The substitution encryption algorithm performs the following steps:

1. Read 'B' from the plain text.
2. Lookup 'B' in the table. Substitute 'B' with 'G'.
3. Write 'G' to the cipher text.

Repeat the above steps for each character in the input text. The resulting cipher text is:

```
GUMTBH
```

The algorithm to decrypt the message is similar to the encryption algortihm.

1. Read 'G' from the cipher text.
2. Lookup the plain text corresponding the the cipher character 'G'. The result here is 'B'.
3. Write 'B' to the plain text.

Repeat these steps for each character in the cipher text to retreive the original message.

Input File Format:

Your program **must** read its input from a file named `input.dat` – use of another input file name will result in a score of zero. The input file contains the substitution table length, the substitution table, and two **or more** commands. Each command is either an encrypt or decrypt operation followed by the text length and the text (which will be cipher or plain text depending on the command). The command, text length, and text may be repeated an unspecified number of times in the input file. In other words, there may be multiple messages to encrypt/decrypt in a single input file. **All commands in the input file use the same substitution table.**

Here is a sample input file:

```
Substitution Table Length -> 5
       Substitution Table -> BOELY._P.:
                  Command -> E
             Text Length -> 8
                    Text -> BYE_BYE.
                  Command -> D
             Text Length -> 8
                    Text -> O.LPO.L:
```

Substitution Table Length

The substitution table length indicates the number of entries in the substitution table. This number is the first line of the input file and occurs in a line by itself.

Substitution Table

The substitution table immediately follows the substitution table length and is of the form: <Plain Char 1><Cipher Char 1><Plain Char 2><Cipher Char 2> ...

Pictorially, the substitution table is as follows:

B	O	E	...	.	:	\n
Plain Character	Cipher Character	Plain Character	...	Plain Character	Cipher Character	New line

With the above input file the substitution table would look like:

Plain Character	***Cipher Character***
'B'	'O'
'E'	'L'
'Y'	'.'
'_'	'P'
'.'	':'

Valid characters that may occur in the substitution table are:

A−Z, a−z	Alphabet
0−9	Numbers
~ ‘ ! @ # $ % ^ & * ( ) _ + | { } : " < > ? − = \ [ ] ; ’ , . /	Symbols and Punctuation
'\n', '\t', '\r'	Special Characters
' '	Space

A specific character appears at most once in the plain text side of the substitution table and at most once in the cipher text side of the substitution table. All characters in the substitution table are below ASCII value 127.

Notice that the substitution table can span lines because it may have a new line character in it. Use the substitution table length to determine the number of characters to read.

Command

The command field is a character. The character is 'E' or 'e' if you are to encrypt the following text. The command character is 'D' or 'd' if you are to decrypt the following text. The command appears on a line by itself.

Text Length

The text length field is an integer that contains the count of the number of characters in the text. The text length is on a line by itself. The text (cipher or plain) begins on the line following the text length.

Text

The text holds the message in cipher text or plain text form. If the command specifies decrypt then the text is cipher text. If the command specifies encrypt then the text is plain text. After the text there is a new line character and then either the end of the file or another command (with a corresponding text length and text). Note that the text (cipher or plain) can span multiple lines. The text will span multiple lines if it has new line characters. You must use the text length to determine when to stop reading characters for the text.

Output File Format:

Your program **must** create an output file called `output.dat`. The first line should contain your name. The second line of the output should be "`Substitution Cipher`". The third line should be blank. The following lines will be dependent on the input file. These lines should contain the operation performed, the output text length, and the output text.

Here is a sample output file (based on the sample input file above):

```
          Title ->Substitution Cipher
     Programmer ->Programmer: Jason Zwolak
     Blank Line ->
      Operation ->Operation: Encrypt
Output Text Length ->8
    Output Text ->O.LPO.L:
      Operation ->Operation: Decrypt
Output Text Length ->8
    Output Text ->BYE_BYE.
```

Operation

This line should indicate the operation that your program performed on the input text to get the output text. This line is either "`Operation: Encrypt`" or "`Operation: Decrypt`".

Output Text Length

The output text length is an integer that contains the number of characters in the output text. It is the same number used in the corresponding Text Length field in the input file. The output text starts on the line following the output text length.

Output Text

The output text should be either cipher text (if the operation is encrypt) or plain text (if the operation is decrypt). The output text should be generated by your program based on the input text using the substitution table. The output text may contain new lines. After your program outputs the last character in the output text, your program should output a new line. For every command, text length, and text in the input file there should be a corresponding operation, output text length, and output text in the output file.

Hints/Questions to ask yourself:

- Use an array to hold the substitution table.
- Can you use the plain character as an index into the substitution table? Would that be useful?
- For decrypting text, can you use the cipher character as an index into a different array? Would that be useful?

Be sure to test your program against all input examples provided to you. Use WinDiff to compare your output to the example output. A WinDiff tutorial can be found at:
http://ei.cs.vt.edu/~cs1044/fall.99/srinidhi/winDiffTutorial/winDiffTutorial.html