Sneakers Lab

INTRODUCTION

Encryption is used in many areas these days. Companies selling products over the Web use encryption to hide credit card numbers and personal information from Internet hackers. Government agencies like the National Security Agency (the premier electronic spy organization of the United States) and the FBI use encryption for everything from transmitting secret information across radio waves to encoding sensitive e-mail.

A few years ago, a movie entitled ***Sneakers*** depicted a program which could decode ***any*** encrypted message. The government obviously wanted to get control of the program, because if someone else had it, their top-secret information would be for sale to the highest bidder. The existence of such a program would literally mean "no more secrets." In real life, such a program could not be written. But you'll get to write your own decoding program in this lab which will allow you to send and receive encoded messages.

In this lab, you'll be writing a program called DECODE.CPP which allows a person to decode a secret message generated by ENCODE.EXE, a program which has been created for you.

The goal of this lab is to increase your array programming skills. You'll also learn how to manipulate characters like integers and how to convert a lower case character into upper case.

ENCODE

The program to encode a secret message (ENCODE.EXE) resides on my website. You may run this file directly from Windows Explorer by double-clicking on it. Run this program to encode a message you'd like to decode later on using your DECODE.CPP program.

There are a few rules which must be followed when typing in sentences:

1. The sentence must contain all letters (upper or lower case).
2. Use underscores ("_") for spaces.
3. Terminate the sentence with the percent ("%") character.
4. The sentence must be no longer than 100 characters.

After entering a "key" and typing in a sentence, ENCODE will convert every letter to upper case and encode the sentence with the specified key. You must remember this key in order to decode the message later on. This key is simply a number between 1 - 25 that is used as an off-set for each letter in the sentence. A key of 2 would mean that each character in the sentence should be incremented by 2 letters. Letters at the end of the alphabet would simply wrap around back to the beginning. Thus A would become C and Z would become B.

For example, if you used a key of 18 and entered the sentence:

```
Ask_not_what_your_country_can_do_for_you%
```

key: 18 the encoded version would read:

```
SKC_FGL_OZSL_QGMJ_UGMFLJQ_USF_VG_XGJ_QGM%
```

Notice that the % and _ characters are not encoded. Each letter is first capitalized and then the key value was added to each letter. So the 1st letter in the alphabet (A) became the 19th letter (S) in the alphabet. The letter U (21st letter) became the letter M (21 + 18 = 39 – 26 = 13th letter in alphabet).

ENCODE.EXE was created using the function Encode below which will encode a given character array (code), given the key to use and the number of characters in the array (size). The code array is an input/output parameter since it comes into the function the way the user typed the message and returns from the function encoded.

```
void Encode(char code[], int key, int size)
{
  int c;

  for (c = 0; c < size; c++)
  {
    // Don't translate underscore
    if (code[c] != '_')
    {
      // Convert any lower case letters to upper case
      code[c] = toupper(code[c]);

      code[c] += key;

      // If letter goes beyond Z, wrap it back around starting at A
      if (code[c] > 'Z')
        code[c] -= 26;
    }
  }
}
```

The way the message is read from the keyboard is as follows.

```
// Read sentence from user until % encountered
c = -1;
do
{
  c++;
  cin >> sentence[c];
} while (sentence[c] != '%' && c < (MAXCHARS - 1));
```

MAXCHARS is a constant defined to be 100. The array sentence is declared as: `char sentence[MAXCHARS]`. Each character is placed into the sentence array until the % character is entered or 99 characters have been entered.

A few notes of interest about ENCODE:

- The function toupper (called in Encode function) takes a character as an argument and simply returns the capitalized version of the character if it had been a lower case letter. Otherwise the character returned is the same. This can be very useful when writing programs that must accept case insensitive input. The file ctype.h must be included to use toupper.

- Integers are added and subtracted from the character variable sentence[c] in the Encode function. Characters and integers can really be treated in much the same way. When we add or subtract from a character, we're actually adding or subtracting from the ASCII value of the character. The ACSII value for 'A' is 65. The ASCII value for a comma is 44. So if you were to add these two characters together, the result would be the letter 'm' because 65 + 44 = 109, the ASCII value for 'm'.

  ```
  char c = 'A' + ',';
  cout << c;    // Prints the letter m
  ```

  On the other hand, if you ran the following code, the number 109 would be printed because the character addition is

being assigned to an integer:

```
int x = 'A' + ',';
cout << x;    // Prints the number 109
```

It also can be shown that '2' + '2' = 'd'. Thus in the computer science arena, 2 + 2 is not always equal to 4. The ASCII
codes are listed on p. 473-474 of your text.

- Character variables can also be compared using the logical operators. The following if statement would be true:

```
if ('a' < 'b')
```

Yet the following if statement would be false because the ASCII value for 'a' is 97, and the ASCII value for 'B' is 66:

```
if ('a' < 'B')
```

- Spaces cannot be used in the sentence because the statement

```
cin >> sentence[c];
```

looks for the first non-white space character entered by the user. That means all spaces, tabs, and carriage returns (hitting Enter) are ignored for input into the character variable. The reason you may enter several characters at once without hitting enter in between or putting spaces between them has to do with the way the computer reads character input. Unlike integer and float cin >> statements, characters do not have to be separated by a space. When 5 characters are typed, and then the user presses ENTER, the cin >> statement grabs the first character entered. Because our while loop keeps executing until the % character is reached (or 100 characters have been entered), the next call to cin >> will grab the next character that has been entered (if any). In this case, because another character has already been entered (the second character), it's placed into sentence[c]. This will continue until all 5 characters are entered in consecutive order into the array.

DECODE

The DECODE.CPP has been started for you and can be downloaded from my web-sight along with ENCODE.EXE. Copy these files to your hard drive using Windows Explorer (before you leave the lab be sure you have the files in a safe place you can get to them later, your M: drive or a floppy). If you don't remember how to do this, ask the instructor for help.

You must write the following 3 functions to complete this program which will convert the secret code back to its original form. It should operate much like ENCODE except in reverse order.

GetMessage - This function will read the encoded message from the keyboard.
Decode - This function will decode the encoded message. It will work just like the Encode function but in reverse. Underscores should become spaces in the translation.
PrintMessage - This function will print the decoded message to the screen.

Here is an example of how DECODE.CPP should run:

```
--MESSAGE DECODER--

Enter the secret key: 18
Enter the code to be decoded:
skc_fgl_ozsl_qgmj_ugmfljq_usf_vg_xgj_qgm%

The sentence reads:
ASK NOT WHAT YOUR COUNTRY CAN DO FOR YOU
```

Notice that the code may be in lower or upper case. The translated sentence should be in all capitals.

Turn in a printout of your working DECODE.CPP program for grading.

"For there is nothing hid,
except to be made manifest;
nor is anything secret,
except to come to light."
Mark 4:22