SEHH2042 Computer Programming

Tutorial 9 – Arrays (2)

Q1. Write a program **toUpperCase.cpp** that accepts an input string (assume less than 100 characters and no blank space is included) from user, and converts all lower-case letters to upper-case. Finally, the converted string is displayed on the screen.

Hints: Consider the ASCII codes of the characters during the conversion. You may have something like this in your codes:

```
// The conversion, assume letter is the character to be changed to upper case
letter = letter - 'a' + 'A';
```

Sample result:

```
Enter a string: TheQuickBrownFox0123456789
Converted string is: THEQUICKBROWNFOX0123456789
```

Think about the following first:

(1) What header files/library are required?
(2) How many variables/arrays are needed?
(3) What is/are the data type(s) of variable(s)/array(s)?
(4) What are the steps and calculations involved?

Follow the steps below to write the program:

1. Include necessary header files/library and write the main() block.

```
    #include <iostream>
    using namespace std;

    int main()
    {
        // code of subsequent steps

        return 0;
    }
```

Use char array instead string object, so no need <string>.

2. Declare character array to store user input text.

```
    char text[100];
```

User inputs at most 99 char + ‘\0’.

3. Ask user to input a string into array *text[ ]*.

```
    cout << "Enter a string: ";
    cin >> text;
```

No need array index. Characters are stored one by one from text[0], there will be ‘\0’ at the end.

4. Use a for loop to go through the string in array *text[ ]*.

```
    for (int i = 0; text[i] != '\0'; i++)
    {
        // code to convert each character
    }
```

Read up to ‘\0’ only, instead of array size 100.

5. In the for loop body, check if that character is lower case letter. If yes, convert it to upper case.

```
    if (text[i] >= 'a' && text[i] <= 'z')
        text[i] = text[i] - 'a' + 'A';
```

Consider ASCII code in comparison and calculation of conversion.

6. After the for loop, display the converted text in array *text[ ]*.

```
    cout << "Converted string is: " << text;
```

No need array index. All elements from *text[0]* up to '\0' are printed.

Q2. In the following program, write a function **swapString()** that accepts two character arrays as arguments, and swaps the first *n* characters between the two words (*n* is the length of the shorter word).

Note: You should use loop appropriately instead of any <cstring> functions.

```
#include <iostream>
using namespace std;

void swapString(char [], char []);  // prototype

int main(){
      char w1[20], w2[20];

      cout << "Enter two words (separated by space): ";
      cin >> w1;
      cin >> w2;

      cout << "Before swapping, words are:\n";
      cout << w1 << endl << w2 << endl;

      // Function call on swapString
      // Insert your code here

      cout << "After swapping, words are:\n";
      cout << w1 << endl << w2 << endl;

      return 0;
}

// Function definition of swapString
// Insert your codes here
```

Sample result:

```
Enter two words (separated by space): Happy Birthday
Before swapping, words are:
Happy
Birthday
After swapping, words are:
Birth
Happyday
```

Q3. Consider the program below. By using the string manipulation functions in the “cstring” library, complete the main function such that strings in array *msgs* are concatenated, with a newline ("\n") after each string, and stored in the character array *buffer*. The number of characters in *buffer*, as returned by **strlen()**, should be 65.

Hint: `msg[i]` is the address of the i-th row in a 2-D array.

```
#include <iostream>
#include <cstring>
using namespace std;

int main(){
      char buffer[80];
      char msgs[10][15] = {
            "a", "ab", "abc", "abcd", "abcde", "abcdef",
            "abcdefg", "abcdefgh", "abcdefghi", "abcdefghij"
      };

      // Put strings in msgs into buffer
      // Your codes should be inserted here.


      // Print the buffer content
      cout << "buffer is:" << endl;
      cout << buffer;

      // Show the length of buffer, using strlen()
      // Your codes should be inserted here.

      return 0;
}
```

<u>Sample result:</u>

```
buffer is:
a
ab
abc
abcd
abcde
abcdef
abcdefg
abcdefgh
abcdefghi
abcdefghij
Length of buffer is: 65
```

Q4. Complete the program below that reads in 3 messages into the character array ***message[NMSG][MAXLEN]*** using a for-loop statement and **cin.getline()** function. It then calls the user-defined function **printRevese()** to print out the messages one by one in a **last-in-first-out** sequence and the characters in each message are displayed in **reverse order**.

Hint: You may use the function **strlen()** to determine the length of each message.
`message[i]` is the address of the i-th row in a 2-D array.

```
#include <iostream>
#include <cstring>
using namespace std;

const int NMSG = 3;      // (Global) maximum no. of messages
const int MAXLEN = 80;  // (Global) maximum length of each message

void printReverse(char [][MAXLEN]); // prototype

int main()
{
      char message[NMSG][MAXLEN];

      // Your code should be inserted here

      return 0;
}

// Your code for printReverse should be inserted here
```

Sample result:

```
Enter message 0: How are you doing?
Enter message 1: Good morning!
Enter message 2: Good afternoon!

The messages are printed in a last-in-first-out sequence.
Characters in each message are printed in a reverse order.
Message 2: !noonretfa dooG
Message 1: !gninrom dooG
Message 0: ?gniod uoy era woH
```

Note: You may add the following code at the beginning of your source file to suppress the warnings and errors related to <cstring> functions: `#define _CRT_SECURE_NO_WARNINGS`