COMP 1602: Computer Programming II

Lab #5: Character Arrays and C-Strings

Please read the notes on "Character Arrays and C-Strings" (Pages 1-5)
***before** attempting the questions on this lab sheet.*

1. Write a program which requests the user to enter a C-string at the keyboard and then finds and displays the character in the middle of the string. For example,

- if the C-string is "Hello", the program should display the character *l*.
- if the C-string is "Welcome!" the program should display the character *o*.

2. (a) Write a function, *intToString*, with the following heading:

```
void intToString (int n, char str[])
```

which accepts an integer value *n* and a C-string *str* as parameters and stores *n* as a sequence of characters in *str* (terminated by the NULL character).

For example,
If *n* = 5, the characters in *str* should be {'5', '\0'}
If *n* = 90, the characters in *str* should be {'9', '0', '\0'}
If *n* = 125, the characters in *str* should be {'1', '2', '5', '\0'}

NB: You *may* need to use a temporary C-string to store the digits of *n* since they will be obtained in reverse order.

(b) Write a *main* function which calls the *intToString* function with *n* = 5, *n* = 90, and *n* = 125 and displays the C-string corresponding to each value of *n* on the monitor.

3. (a) Write a function, *upperToLower*, with the following heading:

```
char upperToLower (char c)
```

which accepts a char *c* as a parameter and returns the lowercase version of *c* if it is a letter; if *c* is not a letter, it returns *c* (unchanged).

(b) Write a function, *isqual*, with the following heading:

```
bool isEqual (char str1[], char str2[])
```

which accepts two C-strings, *str1* and *str2*, as parameters, and returns *true* if *str1* and *str2* have the same characters in the same positions and *false* otherwise. Two letters are considered the same even if the letters have different cases.

(c) Write a *main* function which requests the user to enter two C-strings at the keyboard and then determines if the two C-strings are equal.

4. (a) Write a function, *reverse*, which accepts two C-strings, *str1* and *str2*, as parameters, reverses the characters in *str1*, and stores them in *str2*. The function has the following heading:

```
void reverse (char str1[], char str2[])
```

(b) Using the *reverse* function and the *isEqual* function from Question 3 or otherwise, write a function *isPalindrome* which accepts a string, *word*, as a parameter and returns *true* if the letters in *word* form a palindrome and *false*, otherwise. A palindrome is a word that reads the same from left to right or from right to left. For example, *racecar* is a palindrome. The function has the following heading:

```
bool palindrome (char str1[], char str2[])
```

(c) Write a *main* function which requests the user to enter five C-strings at the keyboard and then determines if each one is a palindrome.

5. (a) Write a function, *indexOf*, which accepts a C-string, *str*, and a character, *c*, as parameters and returns the index of the first location that *c* appears in *str*. If *c* is not present in *str*, the function should return -1. The function has the following heading:

```
int indexOf (char str[], char c)
```

(b) Write a function, *intersect*, which accepts three C-strings *s*, *t* and *u* as parameters and finds the intersection of *s* and *t* and places the result in *u*. There should be no duplicate characters in *u*. The intersection of *s* and *t* consists of the characters that are present in both *s* and *t*. For example, suppose that:

```
s = {'H', 'e', 'l', 'l', 'o', '\0'}
t = {'W', 'o', 'r', 'l', 'd', '\0'}
```

The intersection of *s* and *t* is:

```
u = {'l', 'o', '\0'}
```

(c) Write a *main* function which requests the user to enter two C-strings at the keyboard and then finds and displays the intersection of the two C-strings.

6. (a) A C-string, *haystack*, contains a special message. The message is enclosed by the tags '<' and '>'. Write a function which takes two C-strings, *haystack* and *message*, as parameters and finds and stores the message in *message*. The function has the following heading:

```
void getMessage (char haystack[], char message[])
```

(b) Write a *main* function which reads all the characters from a file, *input.txt* (use your own data) into a C-string, *str*, and then finds and displays the special message stored in *str*.