CS1010 Programming Methodology
Week 11: Characters and Strings

> Learning is not compulsory…
> neither is survival.
> ~ *W. Edwards Deming*

To students:

- Some programs for this discussion are on the CS1010 website, under the "Discussion" page.

I. Exploration

1. Character constants do not only appear in the form of a single character such as 'A', '8' and '@'. Run the following program **q1.c**:

```
#include <stdio.h>

int main(void) {
   int ch1 = '\062', ch2 = '\x41';

   printf("ch1 = %c; ch2 = %c\n", ch1, ch2);

   return 0;
}
```

   What is the output? Can you deduce the meaning of **'\062'** and **'\x41'**?

2. Run the following program **q2.c** and deduce what **atoi()** function does. You need to include **<stdlib.h>** to use **atoi()**.

```
#include <stdio.h>
#include <stdlib.h>

int main(void) {
   char str[10];
   int value;

   printf("Enter input: ");
   fgets(str, 10, stdin);
   value = atoi(str);
   printf("Value is %d.\n", value);

   return 0;
}
```

   What does **atoi()** convert? Try these inputs: (a) 123, (b) 123 456, (c) 123abc, (d) abc123.

3. (a) Assuming that a username can contain up to 8 characters, Brusco wrote this:

```
char username[8];
 . . .
fgets(username, 8, stdin);
```

What is wrong with Brusco's code (**q3a.c**)?

(b) What will happen if Brusco had written the following code (**q3b.c**)?

```
char fruitname[8];
 . . .
strcpy(fruitname, "pineapple");
printf("%s\n", fruitname);
```

4. Do you see any problem with the following program **q4.c**?

```
#include <stdio.h>

int main(void) {
   char board[2][3] = { {'a','b','c'}, {'d','e','f'} };
   int i;

   for (i=0; i<2; i++)
      printf("%s\n", board[i]);

   return 0;
}
```

5. What is the problem with the following program **q5.c**?

```
#include <stdio.h>
#include <string.h>

int main(void) {
   char *fruit1 = "apple", *fruit2 = "apple";
   char *str1 = "yes", *str2 = "yes";

   fruit1 = str1;
   printf("%s\n", fruit1);

   strcpy(fruit2, str2);
   printf("%s\n", fruit2);

   return 0;
}
```

II. Programming on Strings

6. For each of the following string functions, write a small program to illustrate its use. Refer to Table 8.1 in the reference book or look up the Internet for the purpose of the functions.

   (a) **strcat()**
   (b) **strchr()**
   (c) **strtok()**

7. Write a function **count_nonspace(char str[])** to count the number of characters in **str** that are not white spaces. Do not use strlen() in your program.

   [Optionally, write another version **count_nonspace(char *str)** that uses pointer manipulation, i.e. the function body uses *str instead of str[i].]

8. [CS1010 AY2010/1 Semester 1 Exam Q5]
   Write a function **void convert_string(char str[], char dest[])** that converts **str** into **dest** by adding an asterisk between each letter in **str**. Any blank space in **str** is also replaced by an asterisk.

   You may assume that there is one blank space between two words, and only letters and spaces appear in **str**. You may also assume that **dest** has sufficient space to hold the lengthened string.

   For example, if **str** is

   ```
   The quick brown fox
   ```

   then **dest** will be

   ```
   T*h*e*q*u*i*c*k*b*r*o*w*n*f*o*x
   ```

   The above is an exam question. For this discussion, write a complete program that reads a string with at most 20 characters, and calls the **convert_string()** function. Do not use any string functions other than fgets() and strlen().

9. A **palindrome** is a text that reads the same backward as forward. If we disregard case, then the following are palindromic words: “Madam”, “level”, “roTAtoR”.

   (You may go to this website to find some interesting ones (there are many other sites): http://www.innocentenglish.com/tongue-twisters-anagrams-palindromes/best-palindromes.html. Here, however, we will focus on string without spaces in it.)

   Write a program **palindrome.c** to request from the user a word with at most 20 characters. It then calls a function **isPalindrome()** which returns 1 if the word is a palindrome disregarding case, or 0 otherwise.

   Your program should not create any additional array/string.

10. Modify the program **Unit16_Hangman_v1.c** to **Unit16_Hangman_v2.c** as follows:
    - Program will keep a list of 10 words (or more if you like) and randomly choose a word from this list for the user to guess. Each word is at most 15 characters long.
    - Allow user the option to exit the game or continue another game.