

I. Exploration

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)?

Answer:

name should be declared to contain up to 9 characters, to cater to \0:
`char username[9];`

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

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

Answer:

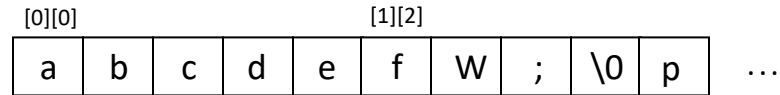
Students may not see what's wrong. They may (most likely actually) get "pineapple" as the output and it seems there is nothing wrong. However, since "pineapple" is more than 8 characters, it spills into "unauthorised" memory space.

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;  
}
```

Answer:

It depends on what values are stored in the memory after the array **board**. Assuming that it is the case below:



Since %s picks up from the starting character specified and ending at a null character \0, the output would be:

**abcdefW;
defW;**

Even if \0 happens to be in the box right after the 'f', the output would be:

**abcdef
def**

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;
}
```

Answer:

Segmentation fault (core dumped)!

The **strcpy()** statement gives a segmentation fault. The **strcpy()** function attempts to copy the characters of the string ("yes") pointed to by str2 into the space pointed to by fruit2. However, fruit2 is pointing to a string literal, which is a read-only space.

II. Programming on Strings

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()`.

Answer: See [convert_string_ans.c](#)

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.

Answer: See [palindrome_ans.c](#)