ilansh
302514401
Ilan Shamir

Part 1:
a. This function inputs the character NULL ('\0') at the end of the given string (str) instead of its last character (assuming the original string ends with NULL).

b. The problem with the function is that the method fsacnf reads words separated by spaces (or other white characters), so the function prints each word scanned from the file in a new line, and also is not limited to a number of characters in each line.
To resolve this problem we can use the method fgets, invoking it with the following syntax:
fgets(nextLine,MAX_LINE_LENGTH,fln)
This call will into the string nextLine from the file pointed by fln, a maximum of MAX_LINE_LENGTH characters, as desired by the function description.
In order to avoid double line spaces (since fgets reads the "end line" character as well), we can remove the "\n" from the following printf command.

c. The code will compile and run properly, but the resulting output would be the string "slabc is an easy course." with some junk characters at the end of it.
The solution to this problem is to exchange the first line from int A[10] to int *A.
The problem was caused by the strcat method, which tries to copy too many characters to the end of the array A, causing unexpected behavior, resulting in the junk letters appended at the end of C (which is the same as A, since C and A point to the same location).
The new code doesn't confine A to 10 characters, and allows for more flexible use by the method strcat.
Another solution would be to simply increase the size of A[10] to A[25].

d. 1. We shouldn't write #include "a.h" in c.c since it is included in b.h, which is already included in c.c.
2. This command wouldn't cause multiple definition of the function func_a because the interface a.h defines the function a.h only if it was not defined before using the command #ifndef A_H.
The first time func_a is defined, it also defines the literal A_H, thus not allowing for multiple definition.
3. The command isn't working properly because there are missing links in the command, and a.c can't find some needed information and definitions.
We can fix the command by typing gcc a.c **-c** -Wall. This will allow us to compile the file a.c without performing the linkage process, outputing the file a.o.
4. The command fails because there are missing files for the linking process, which the file c.c needs in order to work.
A solution to this problem would be to write the command gcc a.c b.c c.c -Wall -o bella.