Technische Universität München

Informatics I for MSE

MSE IN8011
Prof. Bjoern Menze, Giles Tetteh
Philipp Kaeß

WS 19/20
Homework 2

Due Date: 17.11.19, 23:59

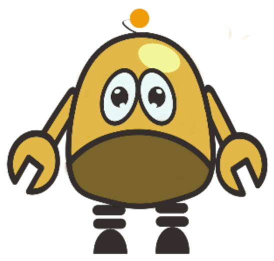
Release Date: 08.11.19

Homework 2 : Socially Awkward Bot

Topics: Strings, Input/Output, Basic `If`/`Else` Statement, Exit, General Function Structure

Introduction

In this homework you will create a program, which is able to communicate with the user in a primitive and predefined way. As the semester has just started, it is possible that some of you may not have found any friends yet, and therefore we will create this artificial chat bot to be their first friend.


Lonely chat bot Boris, 2019, colorized

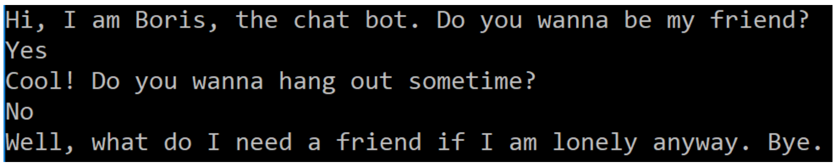
Specifications

When the function `chat_bot()` is called by a main function, the following should happen:

- “`Hi, I am Boris, the chat bot. Do you wanna be my friend?`” is printed.
- The user then has the possibility to answer the bot’s question. If the first letter of the user’s answer is any other than “`Y`” or “`y`”, we consider the answer as a “no” and the bot says “`Oh. Well, then I will ask someone else. Bye.`”, and then the function immediately stops.
- If the first letter of the user’s answer is “`Y`” or “`y`”, the bot says “`Cool! Do you wanna hang out sometime?`”
- Again, the user has the possibility to enter his/her answer. If the answer starts with any other letter than “`Y`” or “`y`”, we consider it as a “no” and the bot says “`Well, what do I need a friend for, if I am lonely anyway. Bye.`”, and then the function stops.
- If the answer starts with “`Y`” or “`y`”, the bot says “`Awesome. See ya!`”, and then the function stops.

Example Output

Your terminal could look like that after an example run of your program:


```
Hi, I am Boris, the chat bot. Do you wanna be my friend?
Yes
Cool! Do you wanna hang out sometime?
No
Well, what do I need a friend if I am lonely anyway. Bye.
```

Tool Box
The explanations below **may** help you to solve this problem.

CHAR-ARRAYs
In C, there actually is no datatype to store strings (e.g. words), there is only the `char`-datatype to store single characters. If we want to store a word, we need to store each `char` individually. Therefore, we use arrays of `char`s, which are basically just a certain number of consecutive `char` variables. We will talk about arrays in-depth in a few weeks, but there are some things you should know already:

- You can create a `char` array of size `n` by calling `char array_name[n] = {0};`
- You can fill this array with input from the user (see INPUT below)
- You can access (e.g. print, compare etc.) single `char`s of your array by `array_name[i]` , where `i` is the index of the `char` you want (starting from 0)
- Every string you store in an array **has to be** NULL-terminated, which means, the `char` after the last `char` of the word has to be a `0`. This ensures, that when you print the word with `%s`, the `printf`-function stops printing after your word. If you use `fgets()` as described below, this happens automatically.

OUTPUT
You can print text and variables with the function `printf()`. To print text, simply put it with double quotes in the brackets. To print a variable (together with text), use a format specifier at the spot where the variable should be printed, and write the variable name after the closing quotes, separated by a comma. To make a new line, use `\n`.
Example for string: `printf("Hi %s, how are you? \n", array_name);`

STRING INPUT
You can get input from the user by calling the function `fgets(buffer, n, stdin)`, where `buffer` is the name of the array where you want to store the input, `n` is the maximum number of characters you want to read from the input (including the terminating 0), and `stdin` is some fancy thing we can't focus on now, so please just use it without asking questions. When you call this function, the flow of the program stops, and the user can enter something and end his/her input by pressing enter. The function then takes the first `n-1` (or less if input has less chars) `char`s of this input and puts it into `buffer`, and then puts a terminating `0` after the last `char`. But: If the users input has more than `n-1` chars, the `char`s that didn't make it into buffer will be stored, and when you call `fgets()` again later in your code, these `char`s will be used before the new user input `char`s. Usually you don't want this, so you can simply "flush" the remaining `char`s by using `fflush(stdin)` after every usage of `fgets`.
Example:
```
char buffer[10] = {0};
fgets(buffer, 10, stdin); //First 9 chars of input are stored
fflush(stdin); //Remaining (if any) chars are deleted
```

IMMEDIATE EXIT
Whenever you want to exit a function, you can use `return`. When dealing with `void`-Functions (such as in this HW), simply call *`return;`* to exit. When a function expects a return value (non-`void`), you need to return a value. (e.g. `return 1;`)

Tasks

We provide you the template file ***lonely_bot.c*** on Moodle. Complete the function `chat_bot()`, so that the program works as specified above. Please note:

- You are only allowed to write code in the spot indicated in the function. Do not change anything else in the function
- **Use exactly the sentences specified above. Don't let the bot say additional things or change the choice of words**
- You are provided with an additional C-File ***main_bot.c***. This file contains a main-Function, which you can use to test your progress. Store the two C-Files in the same directory on your computer, and compile and run the main file. The main-Function will then call your `chat_bot()`-Function
- When you are finished, put **lonely_bot.c** directly in a zip-File named "HW2_Mustermann_Max" (with your name of course) and upload this file to Moodle. Do NOT upload your main-File and do NOT put any folders into the zip-File.

Threshold

To pass this homework, your program has to fulfill the following conditions:

- Your C-File must be compilable without any errors and with the flags -Wall -Werror enabled
- It has to yield the correct output with any possible combination of inputs from {`Yes, yes, No, no`} correctly as specified above
- The bot's answers have to equal the ones given above **exactly**, no words / chars are printed in addition and no words / chars are omitted