Homework #1

Question 1 (3 pt.) Shell

Consider the following main program written in C:

```
int main(int argc, char **argv)
{
        print_args(argc, argv);
        return 0;
}
```

Function `print_args` is a function that returns no value (`void`), and takes the exact same arguments as function `main`. Internally, it traverses the array of input arguments given in `argv`, and containing `argc` strings, and prints it one by one. For the following command, the main program should provide this exact output, where `q1` is assumed to be the program executable:

```
$ ./q1 a b c
argv[0] = './q1'
argv[1] = 'a'
argv[2] = 'b'
argv[3] = 'c'
```

Write the implementation of function `print_args` and save the full source code for this program in a file named `q1.c`. You can compile it using command `gcc q1.c -o q1`, and run it as shown above. Provide file `q1.c` as an attachment to the homework submission on Blackboard, and make sure that this file can be compiled without any changes.

Question 2 (3 pt.) Shell

Write a C program that displays a prompt on the screen (for example, character '`$`'), reads a string from the user, and displays the exact same string when the user presses *Enter*. This process is repeated in an infinite loop, until the user kills the program by pressing *Control+C*.

In order to read the string from the keyboard, use function `fgets`, passing `stdin` as its last argument, indicating that the standard input (keyboard) is the device from which the string should be read. Use the `man` pages to obtain full information about this function.

Notice that `fgets` will keep the *newline* character ('`\n`') corresponding to the *Enter* key at the end of the string. Get rid of it by replacing it with a null character ('`\0`'). Function `strlen` will be useful here. Here is an example for the program output:

```
./q2
$ hello
hello
$ how are you
how are you
$ ^C                              ← Control+C was pressed, program killed
```

Attach your source code in file `q2.c` and upload it on Blackboard. The source file should compile with `gcc` and run correctly on Linux without any modifications.

Question 3 (4 pt.)

Write a C program that creates a child process. The roles of the parent and child process are the following:

- The child process reads an integer value from the keyboard, and returns it as the exit status of the program, either with a `return` statement in the `main` function, or with an invocation to function `exit`.
- The parent process waits for the child process to finish, and prints the child's exit status on the screen, which should be equal to the value read from the keyboard. Read the `man` pages for function `wait` to see the details on how to capture the child's exit status.

This is an example of the program output:

```
$ ./q3
Enter a number: 34                    ← printed by child
Child exited with status 34           ← printed by parent
```

Attach your source code in file `q3.c` and upload it on Blackboard. The source file should compile with `gcc` and run correctly on Linux without any modifications.