Unit 11: The Preprocessor

In previous units we used the `#define` and `#include` directives without going into detail about what they do. These directives and others which are covered in this unit, are handled by the preprocessor, a tool that edits C programs just prior to compilation. The programming language C (along with C++) relies on the preprocessor.
The preprocessor is a powerful tool, but can easily be misused to create programs that are unnessarily difficult to understand. Thus, it should be used in moderation. To use the preprocessor and its directives properly, its important to understand how it works and what are its capabilities.
Relevant for this unit are the following Book Sections:

- Section 14.1: How the preprocessor works
- Section 14.2: General rules that effect all preprocessing directives
- Section 14.3: Macro definitions
- Section 14.4: Conditional compilation

The most frequent use of the preprocessor is the use of so called "guards" in header files and defines of constants that are used in a program. These features are important when writing larger programs which we shall cover in the next unit. In this unit we also look at some other useful applications of the preprocessor for defining specific compilation modes of a program.

Project u11: Sum Debugging Mode

In this project we add a special debugging mode to the sum.c program which allows to print the values of variables of sum.c. It is a special mode because we can specify this mode by defining the variable `DEBUG` either on the command line when invoking the compiler (`gcc -DDEBUG`) or add that define somewhere in the program before the macro `PRINT_DEBUG` ist used. Usually such a debug flag is defined on the command line to the compiler.
Start with adding the header file debug.h to your project with the following contents:

```
#ifdef DEBUG
#define PRINT_DEBUG(n) printf("Value of " #n ": %d\n", n)
#else
#define PRINT_DEBUG(n)
#endif
```

Extend the provided sum.c program such that the name and value of the variables `n` and `sum` is printed if 'DEBUG' is defined and whenever one of the two variables is initialized or assigned a new value. This means you must add above macro call at all relevant positions in the sum.c program.
For example, a solution can be compiled and linked with

- `gcc -DDEBUG u11_sum_d.c -ou11_sum_d`

and tested with

```
% ./u11_sum_d
Value of sum: 0
This program sums a series of integers.
Enter integers (0 to terminate): 1 2 7 3 0
Value of n: 1
Value of sum: 1
Value of n: 2
Value of sum: 3
Value of n: 7
Value of sum: 10
Value of n: 3
Value of sum: 13
Value of n: 0
The sum is: 13

%
```

Note that the new additional output of the program should *exclusively* be produced by `PRINT_DEBUG` and not by any other additional calls of printf.
Refer to the provided test-cases and compare the output of your program with the provided output of the test-cases. Your program should behave exactly like the test-cases before you hand it in. Hand in your program as `u11_sum_d.c`.