C Tutorial 02

01. There are two ways to add comments in C:

    // - Single Line Comment.

    /*... */ - Multi-line Comment.

    Commenting involves placing Human Readable Descriptions inside of computer programs detailing what the Code is doing. Proper use of commenting can make code maintenance much easier, as well as helping make finding bugs faster.

02. The main function

03. 'scanf' allows you to read input from the user or from a file and store that input in variables of different data types.

04. Yes

05. (a) Valid identifier

    (b) Invalid identifier. Identifiers cannot start with a digit.

    (c) Invalid identifier. Identifiers cannot contain hyphens.

    (d) Valid identifier

    (e) Valid identifier

    (f) Valid identifier

    (g) Invalid identifier. Identifiers cannot contain spaces.

    (h) Invalid identifier. Identifiers cannot contain hyphens.

    (i) Valid identifier

    (j) Invalid identifier. Identifiers cannot start with digits and cannot contain spaces or hyphens.

06. a) False. The function ‘printf’ does not automatically begin printing at the beginning of a new line. It continues printing from the current cursor position on the screen unless specified otherwise in the format control string.

b) False. Comments are ignored by the compiler and do not affect the program's execution. They are used to provide explanatory or descriptive text to aid human readers but do not result in any output on the screen.

c) True

d) True

e) True

f) False. C is case-sensitive, so ‘number’ and ‘NuMbEr’ would be considered as distinct and separate variables. The names differ in their casing, and therefore, they are treated as different identifiers.

g) False. It is not necessary for a program to contain a separate ‘printf’ statement for each line of output. Multiple lines of output can be printed within a single ‘printf’ statement by including newline characters (\n) or using separate ‘printf’ statements for each line. The number of ‘printf’ statements depends on the specific requirements and desired output formatting.

07.
```
*
**
***
****
*****
```

08. a) Format specifier for ‘scanf’ should be ‘%d’ to read an integer value.

```
scanf("%d", &value);
```

b) Two errors
* The closing double quote in the ‘printf’ statement is missing.
* The newline character (\n) is placed outside the quotes.

```
printf("The product of %d and %d is %d\n", x, y, product);
```

c) The ‘scanf’ function should start with a lowercase ‘s’, as it is case-sensitive.

```
scanf("%d", &anInteger);
```

d) No errors in the statement.

e) Three errors
* It should be ‘printf’ instead of ‘print’.
* Closing double quote in the format string is missing.
* The comma is placed outside the quotes.

```
printf("The sum is %d\n", x + y);
```

f) Three errors
* The ‘printf’ function should start with a lowercase ‘p’, as it is case-sensitive.
* The closing double quote is missing.
* The symbol ‘&’ should not be used with ‘value’ as it is already a pointer.

09. a) 2

b) 4

c) X=

d) X=2

e) 5 = 5

f) Nothing

g) Nothing

h) Nothing

i) Nothing

10. a) True

b) True

c) False. The statement ‘printf("a = 5;");’ is a ‘printf’ statement, not an assignment statement. It is used to display the value of a variable or other data on the screen. An assignment statement would be something like ‘a = 5;’, where the value ‘5’ is assigned to the variable a.

d) True

e) False. The variable names ‘3g’, ‘87’, and ‘2h’ are invalid. Variable names in C cannot start with a digit. They must start with an underscore or a letter, followed by any combination of letters, digits, or underscores.