Tutorial 02

1) In a C program, comments are denoted by a /* at the start and a */ at the conclusion. As an example:

    ```
    /* This is a comment */

    // This is a single-line comment
    ```

    Comments in a program are used to give explanations and documentation for the code. The compiler ignores comments designed to assist others to understand the code, including the developer who produced it and anybody else who may be reading or working with it. Comments can also temporarily disable a part of the code for debugging or testing purposes.

2) The main() function is critical in every C application. It is the program's starting point and where the program execution begins. The operating system calls the main() method when the application is run. The main() method is unique in C and C++ because the operating system or runtime library looks for it explicitly to begin executing the program.
3) In C, the scanf function is used to read user input and store it in variables. It is included in the standard input/output library (stdio.h). The scanf function accepts as its first parameter a format string, followed by the variables in which the input should be saved. The format string provides the input type and format, such as a decimal integer, text, or character.
4) C is, indeed, a case-sensitive language. This signifies that the language recognizes uppercase and lowercase letters. The variable count, for example, is not the same as the variable Count or COUNT. This holds true for keyword, function, and variable names.
5)
    a. record1 is a valid identifier.
    b. 1record is an invalid identifier because it starts with a number. Identifiers can't start with a number in C
    c. file-3 is an invalid identifier because it contains a hyphen. Hyphens are not allowed in identifier names.

d. return is an invalid identifier because it is a keyword. Keywords are reserved words and can't be used as identifiers.
e. $tax is an invalid identifier because it starts with a special character. Identifiers can't start with a special character such as $, @, &, etc.
f. name is a valid identifier.
g. name and address is an invalid identifier because it contains a space. Identifiers can't contain spaces.
h. name-and-address is an invalid identifier because it contains a hyphen. Hyphens are not allowed in identifier names.
i. name_and_address is a valid identifier.
j. 123 - 45 - 6789 is an invalid identifier. As it contains multiple spaces and hyphens which are not allowed in identifier names.

6)

a. This is incorrect. The printf function does not always start printing at the beginning of a new line. Unless a newline character (n) is provided in the format control string or the cursor is moved to a new line in another way, it will continue to print on the same line.
b. This is incorrect. The compiler ignores comments and does not force the computer to output anything on the screen when the program is executed. They are used to offer code explanations and documentation.
c. True. When the escape sequence n is used in a printf format control string, the cursor moves to the start of the next line on the screen.
d. This is incorrect. Variables can be defined at any time during the process.
e. True. When variables are defined, they must be assigned a type, such as an int, float, char, and so on.
f. This is incorrect. Because C is a case-sensitive language, variables number and NuMbEr are treated as distinct variables.
g. This is incorrect. By including newline characters (n) in the format control string, a program can print three lines of output with a single printf statement.

7) The code will output like this:

```
*
```

```
**
***
****
*****
```

8)

a. The first error in the statement is in the scanf function's format string; the d should be replaced with %d to read an integer value; the second error is the absence of the address operator & before the variable value to provide the variable's memory location to the function.

b. The first error in the statement is that the newline is not included within the quotation marks, and the second fault is the missing); at the end.

c. The first error in this statement is the capitalization of the scanf function, C is a case-sensitive language, so the function should be written in lowercase.

d. The fault in this line is that the printf function contains an additional variable in the format string, which causes an error during program execution because the function only expects two variables but receives three.

e. The first error in this statement is the function name, it should be printf, not print and the second error is the missing operator + between %d\n, and x, also the missing) at the end of the statement.

f. The first error in this statement is the function name, it should be printf, not Printf, the second error is that the variable value should not be preceded by the address operator & and the missing) at the end of the statement.

9)

a. The statement will print the value of x, which is 2.

b. The statement will print the value of x+x, which is 4.

c. The statement will print the string "x=".

d. The statement will print the string "x=" followed by the value of x, which is 2.

e. The statement will print the value of x+y=5 and y+x=5

f. This is an assignment statement, it will assign the value of x+y to the variable z, but it won't print anything.

g. This is a function call statement, it will read the input from the user and store it in the variables x and y but it won't print anything.

h. This statement is commented out, it will not be executed by the compiler and it won't print anything.
i. The statement will print a newline character (\n)

10)

a. True. C operators are evaluated from left to right depending on operator precedence and associativity. This means that higher precedence operators are assessed first, whereas operators with the same precedence are evaluated from left to right.
b. True. Valid variable names include: under the bar, m928134, t5, j7, her sales, his account total, a, b, c, z, z2. They all adhere to the C naming standard, which states that variable names must begin with a letter or an (underscore) and include only letters, digits, and_.
c. False. The statement printf("a = 5;"); is not an assignment statement, it is a function call statement that will print the string "a = 5;" on the screen. An assignment statement should look like this a = 5;
d. True. A valid arithmetic statement with no parentheses is evaluated from left to right depending on operator precedence and associativity.
e. True. All of these are incorrect variable names because they begin with a number, and variable names in C cannot begin with a number.