1.

// This is a single-line comment

 Comments can be used to explain what the code does, why the code is written in a certain way, or to provide any other information that might be helpful to the reader of the code.

2.

 printf() function is used to print formatted output to the console.

3. The purpose of scanf is to read input values and store them in variables. It takes a format string as its first argument, which specifies the expected format of the input. The

4. Yes, the "standard C" programming language is case sensitive. This means that distinctions are made between uppercase and lowercase letters in identifiers (variables, functions, keywords, etc.) within the C language.

5.

Valid =

record1, return, $tax:, name: name, and, address:

invalid =

1record, file-3, name and address, 123-45-6789

6.

a) False. The `printf` function in C does not automatically begin printing at the beginning of a new line. It continues printing from the current cursor position on the same line unless specified otherwise.

b) False. Comments in C, denoted by `/\*` and `\*/`, are ignored by the compiler and do not affect the execution of the program. They are used for providing explanatory notes or disabling code temporarily. The text enclosed between `/\*` and `\*/` is not printed on the screen when the program is executed.

c) True. The escape sequence `\n` is used in C's `printf` function to insert a newline character. When used within a `printf` format control string, `\n` causes the cursor to position at the beginning of the next line on the screen.

d) False. In C, it is not required to define all variables before they are used. However, variables should be declared before they are used to inform the compiler about their existence and type. The actual definition (assignment of initial value) can occur later in the code.

e) True. In C, all variables must be given a type when they are defined. The type defines the nature of the variable, such as integer, floating-point, character, etc., and determines the memory allocated to it and the operations that can be performed on it.

f) False. C is case sensitive, so the variables `number` and `NuMbEr` would be considered as distinct and separate variables with different names.

g) False. A program that prints three lines of output does not necessarily require three `printf` statements. It could use a single `printf` statement with appropriate formatting, including escape sequences like `\n` to introduce line breaks within the output. Multiple lines of output can be generated with a single `printf` statement.

7.

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

8. a) Error: The format specifier in `scanf` is missing the `%` symbol before the `d`.

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

b) Errors: Missing a comma at the end of the format string. The closing double quote should be placed after the comma.

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

c) Error: The function `Scanf` is written with an uppercase 'S'. C is a case-sensitive language, so the correct function name should be `scanf`.

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

d) No errors found.

e) Error: The function `print` is undefined in C. The correct function for printing is `printf`. The comma is missing between the format string and the argument list.

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

f) Error: The function `Printf` is written with an uppercase 'P'. C is a case-sensitive language, so the correct function name should be `printf`. The ampersand symbol should not be included before `value` since it's not necessary for `printf`.

Correction: `printf("The value you entered is: %d\n", value);`

9.

a) printf("%d", x); prints "2".

b) printf("%d", x + x); prints "4".

c) printf("x="); prints "x=".

d) printf("x=%d", x); prints "x=2".

e) printf("%d = %d", x + y, y + x); prints "5 = 5".

f) Nothing is printed because it is an assignment statement (z = x + y;), not a printing statement.

g) The scanf function reads input from the user, so it waits for input. Nothing is printed immediately unless the user provides input.

h) The line is commented out, so nothing will be printed.

i) printf("\n"); prints a newline character. So, it prints a newline (blank line).

10.

a) True. In C, operators are evaluated from left to right. This means that in an expression with multiple operators of the same precedence level, the operators are applied in the order they appear from left to right.

b) True. Each of the provided variable names is a valid variable name in C. Variable names can include letters (both uppercase and lowercase), digits, and underscores. However, they cannot start with a digit.

c) False. The statement `printf("a = 5;");` is not an assignment statement but a `printf` statement that will print the text "a = 5;" as it is. An assignment statement assigns a value to a variable, such as `a = 5;`, where the variable `a` is assigned the value 5.

d) True. In C, a valid arithmetic expression containing no parentheses is evaluated from left to right. The operators in the expression are applied in the order they appear, following the precedence and associativity rules of the operators.

e) True. The variable names "3g", "87", "67h2", "h22", and "2h" are all invalid variable names in C. Variable names cannot start with a digit; they must start with a letter or an underscore.