1. Match the different parts of the program below

<https://github.com/carethik2k/C_Sharp_Class/blob/master/Exam_1/Question_1/Program.cs>

|  |  |
| --- | --- |
| Question\_1 | Namespace |
| public Demo() | Constructor |
| public Demo(int testValue) | Parametrized Constructor |
| \_testVar | Field |
| TestVar | Property |
| Program | Class |
| objDemo | Object / Instance |
| using System; | Directive |

1. What is the default value for the following data types.

|  |  |
| --- | --- |
| Bool | False |
| String | Null |

1. What is the difference between a ‘local variable” and a ‘field’ ? Explain in a few words (or) show the difference using a simple program.

Local variable :

* Variables represent storage locations. Every variable has a type that determines what values can be stored in the variable.
* A variable created in a function destroyed as soon as it loses the scope.
* Local variable can be given by an implicit type of "var" instead of explicit type.  
      eg: var i = 10  //implicitly typed  
            int j = 10  // explicitly typed
* Below are the restrictions of implicit type variables:

a) "var" can only be used when a local variable is declared and initialized in the same statement.  
    b) "var cannot be used on fields at class scope.  
    c) "var" cannot be used as an initialization expression.

* There are three type of variables in c#. They are "object", "dynamic" and "var". Here var is more type safe. "object" variable provide little information about the type to compiler. "dynamic" variable will not provide any information about the type to compiler.

Field :

* A field is a variable of any type that is declared directly in a class or struct. Fields are members of their containing type.
* Fields can be marked using Access Modifiers like public, private, protected, internal etc.
* Field can declare as static. No instance of the class is necessary to access the static field.
* A field can be declared read only. A static read only field is very similar to a constant.
* A field destroys when the scope of the class is destroyed. In case of Static variable, it only destroyed after all instances of the class/ struct is destroyed.

eg. private string nameOfStaff;

private const int hourlyRate = 0;

1. Consider an integer variable ‘x’ in a program. What is the difference between ++x and x++.

++x :

* The pre-increment operators increment their operand by 1, and the value expression is the resulting incremented value.
* The pre-increment operator "plus plus x" adds one, and returns the **new** value:

int x = 1;

int y = ++x; // now the value of x is 2 and y is 2

x++ :

* The post-increment operators increase the value of their operand by 1, but the value of the expression is the operand's original value prior to the increment operation.
* The post increment operator "x plus plus" adds one, and returns the **old** value:

int x = 1;

int y = x++; // now the value of x is 2 and y is 1

1. Consider the following statement in a program. Explain what happens in this statement and what is the value of x after the execution of this statement.
2. x -= 5;
3. int x = (int) 2.19;

x -=5 :

* “-=” sign is actually a shorthand that combines the assignment sign with the subtraction operator. Hence, x -= 5 simply means x = x-5;
* If we want to decrement ‘x’ by 5, the program will first evaluate the expression on the right (x-5) and assign the answer to the left.

eg: if input = 10;

x = x-5;

then x = 5;

int x = (int) 2.19 :

* Type casting is to convert one numeric data type to another. Here for instance, we can cast a non integer to an integer because it is necessary to convert from one data type to another, such as from double to an int.

int x = (int) 2.19;

* After conversation the decimal portion is truncated.
* After execution the value of x will be 2.

6. Explain the difference between value type and reference type.Explain in a few words (or)

and show the difference using single program.

Value type:

* A data type is a valuetype if it holds the data within its own memory allocation. Value types include the following:
* All numeric data types
* Boolean, Char, and Date
* All structures, even if their members are reference types
* Enumerations, since their underlying type is always SByte, Short, Integer, Long, Byte, UShort, UInteger, or ULong
* Every structure is a value type, even if it contains reference type members.

eg. int num =5;

Variable ‘num’ stores the actual value 5;

Reference type:

* + - A reference type contains a pointer to another memory location that holds the data. Reference types include the following:
  + String
  + All arrays, even if their elements are value types
    - * Class types
      * Delegates
    - A class is a reference type.

eg. string input = “hello”;

Variable ‘string’ does not store ‘hello’. Instead it is created and stored elsewhere in computer memory.