Part01

1. Why does defining a custom constructor suppress the default constructor in C#?
   Compiler understands that I want to change the default logic and put my own, and if it let the default constructor it allow to skip the logic I have made.

2. How does method overloading improve code readability and reusability?
   The same method can be used with different parameters depending on the way you call you will get the appropriate method and it called automatically depending on the number or type of parameters.

3. What is the purpose of constructor chaining in inheritance?
   To reuse the constructor logic and avoid code duplication.

4. How does new differ from override in method overriding?
   - New: Create a completely different method and don't have reference on the old one.
   - Override: Still have reference on the old method and add new features.

5. Why is ToString() often overridden in custom classes?
   To reuse it to have meaningful output, it's original output is the datatype of class attributes that has no meaning.

Part02

1. What is the difference between class and struct in C#?
    - Class is reference type but struct is value type.
    - Class is stored in stack as reference and it's valued in heap meanwhile the Struct is stored in stack only
    - Class can perform inheritance but struct can't.
    - Struct cannot create default constructor automatically, class can.
    - Struct is public by default, but class is internal by default.
    - Class has limitless features, struct is limited.
    - Class is larger in size and used in large systems, the struct is the opposite
    - Struct is cheaper in allocation than class.
    - Class must use new keyword in creatin new variable, struct doesn't have to use it.
    - In class function member can be abstracted, struct not.
    - In class tow variable can have the same reference and changes in one affect the other, struct every variable has it's own copy of data and doesn't affect the other variables.

2. If inheritance is relation between classes clarify other relations between classes
    - Association: "has a" relation between to classes but there is no dependency
    - Aggregation: class has the other but still no dependency.
    - Composition: class has the other and depend on it.
    - Dependency: class need the other as a parameter for example.