1. **Что такое CIL код?**
   CIL – это один из языков программирования .NET. При создании сборок .NET с помощью выбранного управляемого языка соответствующий компилятор всегда транслирует исходный код в инструкции CIL. CIL код можно компилировать благодаря утилите ilasm.exe. Как мной было прочитано CIL язык - это ассемблер на стероидах.
2. **Для чего нужна утилита peverify.exe?**
   Если утилита ilasm.exe необходима для компиляции сборки, т.е. для проверки синтаксиса, то утилита peverify.exe полезна для проверки её семантики. Проверяет достоверность всех кодов операций CIL внутри указанной сборки.
   Пример: стек вычислений должен всегда быть пустым перед выходом из функции, если это не так утилита peverify сообщит об этом.
3. **Что такое возвратное проектирование?**
   Это процесс дизассемблирования сборки .NET, ее анализ, умение редактировать CIL код и заново компилировать в сборку. Возвратное проектирование необходимо в случаях когда исходный код сборки утерян или не доступен, необходимо изменить неэффективный или не корректный CIL код, надо сконструировать библиотеку взаимодействия с COM и учесть ряд атрибутов COM IDL.
4. **Что означает все, что начинается с точки в коде CIL и без нее?**
   Все что начинается с точки в CIL коде является директивой, заставляя выполнять некоторое действие, такое как создание функции или класса. Все что начинается не с точки является инструкцией в CIL коде. Директива CIL позволяет информировать компилятор CIL о том, как должны определяться пространства имен, типы и члены, входящие в состав сборки.
5. **Как ilasm понимает с какой функции необходимо начать работу?**
   В отличии от языков высокого уровня, в которых точка входа является функция main, в CIL коде это не так. Для того что бы компилятор знал с какой функции надо начать работать в нее надо поместить директиву .entrypoint. Поэтому входная функция может иметь любое имя, как например в коде CarClient.il такой функцией является FOOBAR.
6. **Для чего необходима в CIL коде директива assembly?**
   Директива .assembly применяется для перечисления внешних сборок, которые будут использоваться текущей сборкой. Для этого директиву .assembly необходимо пометить атрибутом extern <имя сборки>. Внутри такой директивы должны быть директивы .publictoken и .ver (если сборка со строгим именем). Так же assembly используется для указания нашей сборки. Пример: .assembly <имя нашей сборки>
7. **Для чего нужна инструкция ldstr?**
   Эта инструкция помещает в стек строку, а стек необходим для упрощения передачи параметров в функцию, т.е. все функции получают свои параметры из стека. Также для загрузки в стек используется инструкция ldarg.
8. **Как в CIL коде писать ООП приложение?**
   Для создания ООП приложения используется директива .class. Эта директива снабжается дополнительными атрибутами (такие как public, auto, ansi, extends) и как в C# всегда явно или не явно тип будет унаследован от базового класса System.Object. Т.е. для создания ООП приложения на CIL мы должны как минимум использовать .class и статический .method директивы.
9. **Как в ООП приложении на CIL пишется конструктор?**
   Так же как и в ЯП высокого уровня можно не писать конструктор, но если мы решили написать конструктор на CIL должно выполняться ряд правил. Нужно определить специальный метод, который называется .ctor с такими атрибутами как specialname, rtspecialname и instance. Если класс расширяет другой класс, то нужно определить конструктор который вызовет конструктор родителя.

10. Как будет выглядеть код CILCars.il и CarClient.il на C#?

В CILCars.il в одном пространстве имен написано два класса, поэтому этот файл разделяется на два C# файла: CILCar.cs и CILCarInfo.cs.

CILCar.cs:

```
using System;

namespace CILCars
{
  public class CILCar
  {
    public int currSpeed;
    public string petName;

    public CILCar(int c, string p)
    {
      this.currSpeed = c;
      this.petName = p;
    }
  }
}
```

CILCarInfo.cs:

```
using System;
using System.Windows.Forms;

namespace CILCars
{
    public class CILCarInfo
    {
      public static void Display(CILCar c)
      {
        string caption = string.Format("{0}'s speed is:", c.petName);
        MessageBox.Show(c.currSpeed.ToString(), caption);
      }
    }
}
```

CarClient.cs

```
using System;
using CILCars;

namespace CarClient
{
  class Program
  {
    static void Main(string[] args)
    {
      CILCar c = new CILCar(55, "Junior");
      CILCarInfo.Display(c);
    }
  }
}
```