**C# Homework 14**

**Question 1**

What is the difference between a managed resource and an unmanaged resource?

**Answer**

Managed resources are those that are pure .NET code and managed by the runtime and are under its direct control. Unmanaged resources are those that are not. File handles, pinned memory, COM objects, database connections etc.

**Question 2**

When is memory for an object (reference type) allocated? When is the memory deallocated?

**Answer**

The memory space is allocated to the data members of a class only when an object of the class is declared, and not when the data members are declared inside the class. A variable is deallocated when the system reclaims the memory from the variable, so it no longer has an area to store its value. For a variable, the period of time from its allocation until its deallocation is called its lifetime. The most common memory related error is using a deallocated variable.

**Question 3**

What is a destructor?

**Answer**

In object-oriented programming, a destructor is a method which is invoked mechanically just before the memory of the object is released.

**Question 4**

What is the difference in syntax between a constructor and a destructor?

**Answer**

A constructor is generally used to initialize the data members of the class, whereas a destructor is used to let the object perform some action before it is destroyed.

**Question 5**

Give some examples of scarce resource. Why would you want to manage scarce resources?

**Answer**

Scarce resources such as memory, database connections, or file handles need to be released, and they need to be released as soon as possible. In these situations, your only option is to release the resource yourself.

**Question 6**

What is exception-safe disposal?

**Answer**

One way to ensure that a disposal method (such as Close) is always called, regardless of whether there is an exception, is to call the disposal method within a finally block.

TextReader reader = new StreamReader(filename);

try

{

string line;

while ((line = reader.ReadLine()) != null)

{

Console.WriteLine(line);

}

}

finally

{

reader.Close();

}

**Question 7**

How do you think that the using statement works for resource management? Give an informal, English language, explanation of how it works.

**Answer**

The using statement allows the programmer to specify when objects that use resources should release them. The object provided to the using statement must implement the IDisposable interface. This interface provides the Dispose method, which should release the object's resources.

**Question 8**

What ill effects could result from attempting to dispose of a resource more than once?

**Answer**

A method implementation contains code paths that could cause multiple calls to System.IDisposable.Dispose or Dispose equivalent, such as a Close() method on some types, on the same object. A correctly implemented Dispose method can be called multiple times without throwing an exception. However, this is not guaranteed and to avoid generating a System.ObjectDisposedException you should not call Dispose more than one time on an object.

**Question 9**

We will look at threads later in the term. For now, what is your understanding of how threads interact with resource management? A good guess is a sufficient answer to this question.

**Answer**

A process is an executing program. An operating system processes to separate the applications that are being executed. A thread is the basic unit to which and operating system allocates processor time. Each thread has a scheduling priority and maintains a set of structures the system uses to save the thread context when the thread’s execution is paused. The thread context includes all the information the thread needs to seamlessly resume execution, including the thread’s set of CPU registers and stack. Multiple threads can run int the context of a process. All threads of a process share its virtual address space. A thread can execute any part of the program code, including parts currently being executed by another thread.