**Part III**

**Short answer questions**

1. **Explain the concept of exceptions and exception handling, when they are used and how to catch exceptions.**

Exceptions is use try and catch constructs to catch and handle exceptions.

For example, when the program needs use StreamReader to read a csv file, there are potential issues, then we use try and catch exception. StreamReader try to read the csv file, if nothing happen will be writeline to display the contain, otherwise to catch which issue are there. Such as the file not found or the file cannot open etc.

1. **Differentiate between ‘class’ and ‘struct’ in C#.**

* A class is a reference type, a struct is a value type.
* Class instances are called object and live on the heap, struct instances are called values and live on the stack.
* Class can declare a default constructor, struct cannot.
* After declare constructor, class will not complier still generate the default constructor, but struct can.
* In the constructor, class can initialize a filed automatically, but struct cannot.
* Class can initialize instance field at their point of declaration, but struct cannot.

1. **Explain the possible reason a class in C# can only inherit from a single class, while being able to implement multiple interfaces. Explain the sequence of constructors that will be called when you create an object that is an instance of derived class.**

When Class B and Class C inherit from Class A, and Class D inherits from Class B and C. If there is a method in A that B and C have overridden, and D does not override it, then which version of the method does D inherit, that of B or C. This is a big problem.

The base class constructor is called first.

1. **Write short note on Big-O Analysis of Algorithms.**

Big-O analysis algorithms defines the upper limit, which only limits the function above. For example, the case of insertion sort will take liner time of the best or quadratic time in the worst.

1. **Explain briefly dynamic programming and backtracking.**

Dynamic programming is a way to structure an algorithm such that it divides a bigger problem into smaller subproblems and avoid ever recomputing the same subproblem twice.

Backtracking is usually a recursive way to do a depth first search for a solution in a tree of possible solutions, where the call stack is a store of nodes of the tree which have unvisited branches. The subtrees are pruned optimally, so a massive solution space is reduced to a very small one.

If the branches of the tree overlap, dynamic programming may be improved some backtracking problems, because it will repeat previous computing.