1. 1 Normal form : every column should contain, atomic values
2. 2 Normal form: There should be no partial dependency
3. 3 Normal form: No transitive dependency

* Multi-Threading :
  + Thread
  + Task
  + Thread Pool
* Memory Management:
  + Garbage Collections
* LINQ:
* Design Pattern:
  + Abstract factory
  + Singleton pattern
  + Decorator Pattern
  + Adapter pattern
* SOLID design Pattern
  + Single Responsibility
  + Open- closed Principle
  + Liskov Principle
  + Interface Segregation
  + Dependency Inversion
* Oops concepts
* Delegates, events, Function, Actions

Sites for preparation:

1. **High Scalability**

Algo Expert

**Lucene:**

IndexWriter:

IndexSearcher:

WordNet java library to get The synonyms from Princeton University

HindiWordnet from IIT Bpombay

How indexing is done:

How searching is done:

**Real Life applications of Data structures:**

**Stack:**

1. Undo any operations
2. TO go to the previous page in the browser
3. To Solve expression
4. Backtracking
5. Recursion
6. Memory Variable

**Binary Search Tree:**

1. Used for managing Virtual Memory Areas
2. Used for indexing IP Addresses
3. Used to evaluate Expression trees

**HashTable:**

1. Mac Address of the Netwrok switched
2. DNS Lookup
3. Address of object in the memory
4. Location of the house or something
5. License of some software

**Binary Tree:**

1. Implementing Routing table in router
2. Data Compression code
3. Implementing Expression parser and expression solver
4. To solve database problem such as indexing
5. Expression evaluation
6. To create the hierarchical folder structure

**Queue:**

1. Person at the ticket counter
2. Car at the toll plaza
3. Language Checking Machine
4. Phone answering System
5. Queue of the process in Operating System

**Multi-Threading:**

**Amazon Interview:**

**Managerial Round Questions:**

1. Discussed my current profile and projects in my current company.
2. Why I want to leave my current company.
3. Long Discussion about my projects.
4. One of my projects was to develop a notification service.
5. He found it of his interest and asked for a detailed explanation. He asked the difference between a service and program.
6. What is thrashing?
7. My 3 most challenging tasks I have worked on.
8. My 3 most boring tasks I have worked on.
9. My 3 strengths.
10. My 3 weaknesses.
11. Do you have ever faced a situation in which you have to stretch yourself?
12. As a child how did you see your future?
13. What is your favourite data structure and why?
14. What is O(n) and what’s its use?
15. Will you be able to relocate to Chennai?/ Do you have issues with relocation?

**Bar Raiser Round:**

1. Discussed my current profile and projects in my current company.
2. Do you have issues with relocation? He told me a lot of candidates come here and then try to relocate back to other locations.
3. He took a deep dive into one of my projects and asked me questions like
4. What challenges you faced while doing this project and how you resolved them.
5. Why do you want to join Amazon?

**Ref Keywords in C#:**

* An argument that is passed to ref or in parameter must be initialized before it passed.
* For Out parameter it is not necessary to initialized before they are passed.
* Members of class can’t have signature that are differ only by ref, out , in.
* Methods can be overloaded when one method has a ref, in , out parameter and other has a value parameter.
* You can’t use the ref , in or out keywords for the following kind of methods:
  + Async Methods
  + Iterator methods, whicj include a yield return or yield break;
* Passing a reference type reference enabled the called method to replace the object to which the reference parameter refers in caller.

**Params Keyword:**

* By using the params keyword, you can specify a method parameter that takes a variable number of arguments.
* You can send a comma-separated list of arguments of the type specified in the parameter declaration or an array of arguments of the specified type. You also can send no arguments. If you send no arguments, the length of the params list is zero.
* No additional parameters are permitted after the params keyword in a method declaration, and only one params keyword is permitted in a method declaration.
* The declared type of the params parameter must be a single-dimensional array,

**INTERFACE & ABSTRACT CLASS:**

think you should use interfaces when you want a full implementation and use abstract classes when you want partial pieces for your design (for reusability)

* **Consider using abstract classes if any of these statements apply to your situation:**
  + You want to share code among several closely related classes.
  + You expect that classes that extend your abstract class have many common methods or fields, or require access modifiers other than public (such as protected and private).
  + You want to declare non-static or non-final fields. This enables you to define methods that can access and modify the state of the object to which they belong.

**Consider using interfaces if any of these statements apply to your situation:**

* You expect that unrelated classes would implement your interface. For example, the interfaces Comparable and Cloneable are implemented by many unrelated classes.
* You want to specify the behavior of a particular data type, but not concerned about who implements its behavior.
* You want to take advantage of multiple inheritance of type.

Nullable type:

* Use the null-coalescing operator, ??, to assign a value to an underlying type based on a value of the nullable type: int? x = null; int y = x ?? -1;. In the example, since x is null, the result value of y is -1.

**Why is IL (Intermediate Language) is half compiled:**

* So that during runtime we the compiler is able to figure out the operating system, hardware configuration and compile a optimal code as per the environment
* JIT compiles code in 3 ways
  + Per file
  + Per Method\function
  + Code fragment
* Type of Git
  + Normal JIT: compilation done dynamically(when methods encounter), it store compiled code in cache
  + Econo JIT: same as Normal JIT, but doesn’t store code inmemory
  + Pre-JIT: Full Compilation before the code run

**Read Only & Constant:**

* Constant is compile time constant and Read Only is runtime constant
* Constant values can’t be changed, read only variable can be changed in Constructor

# WPF tutorials

Type of Layouts in wpf:

There are five types of panel . All these layouts are derived from panel layout

* Stack panel:
  + Stack panel is simple container control that just shows one element after another. The orientation of the stack panel can be horizontal or vertical.
* Wrap Panel:
  + Wrap panel position children from left to right . one after the another as long as they fit into the line and then it continues from the next line. The panel orientation can be from horizontal or vertical.
* Canvas
* Dock Panel
* Grid

**Dependency Property:**

* Dependency properties are basis for several key WPF feature including data binding, animation and styles.
* It also provides change notification
* This is the syntax of the dependency property

public int MyProperty

{

get { return (int)GetValue(MyPropertyProperty); }

set { SetValue(MyPropertyProperty, value); }

}

// Using a DependencyProperty as the backing store for MyProperty. This enables animation, styling, binding, etc...

## public static readonly DependencyProperty MyPropertyProperty =

## DependencyProperty.Register("MyProperty", typeof(int), typeof(ownerclass), new PropertyMetadata(0));