**1) You are building a help ticket system. You want to ensure that the older the ticket, the sooner it will  
be handled by the team. For example, a ticket submitted a week ago has a higher priority than a  
ticked just submitted. What kind of data structure would you use, and why?**

-In this instance I would implement a queue data structure because it employs a first-in-first-out (FIFO) model.

**2) You are building a tracking system for seasonal agricultural farm labor. The labor requirements vary  
widely, depending on the season. Your requirement is that the newest hires are terminated first, and  
that our more experienced hires are kept longer. What kind of data structure would you use, and why?**

-In this instance I would implement a stack data structure because it employs a last-in-first-out (LIFO) model.

**3) You are building a transaction database. Your requirement is that the database adds data very quickly, and that deletions, updates, and searches happen infrequently. In other words, data is typically added in the order in which the transaction occurs. What kind of data structure would you use, and why?**

-In this instance I would implement a list because it offers the functionality posed in the question.

**4) You are building an analytical database. Your requirement is that the database handle queries very  
quickly, but that the data never changes, i.e., there are no inserts, deletions, or updates. What kind  
of data structure would you use, and why?**

-In this instance I would implement a dictionary collection class because it creates an index that maps to each element which would allow for querying and would also handle all the other requirements posed in the question.

**5) You are building a personnel directory, where searched are performed by last name, first name, middle name. What kind of data structure would you use, and why?**

-In this instance I would implement a SortedList collection because it allows for the assignment of keys to associated values. These keys allow for the querying by such information as last name, first name, and middle name. Additionally, querying can be done faster because a sorted list is stored alphabetically.

**6) You are building a username/password database. Your requirement is that updates happen frequently (when users change their passwords) and that searches (to authenticate users) happen extremely quickly. What kind of data structure would you use, and why?**

-In this instance I would implement a HashSet collection class because it supports the rapid searching of data and supports the ability to cross check information through the utilization of the *IntersectWith, UnionWith, and ExceptWith* methods.

**7) What is a lambda expression? Give an example. Why would we use a lambda expression?**

-A lambda expression is an expression that is used to create an anonymous function. You would use a lambda expression in any code that requires instances of delegate types or expression trees. For example, as an argument to the Task.Run(Action) method to pass the code that should be executed in the background. You can also use lambda expressions when you write LINQ in C#. An example of a lambda expression is list.Find(p => p.LastName);

**8) What is the difference between lambda expressions and anonymous methods? What are the advantages of each?**

-The difference between lambda expressions and anonymous methods is that anonymous methods are essentially functions without a name with the ability to create closures. Lambda expressions are constructs that are convertible to both anonymous methods and expression trees. They also follow more complex rules of type inference than anonymous methods.

Lambda expressions supersede anonymous methods as the preferred way to write inline code. There is one case in which anonymous methods enable you to omit the parameter list, which means that an anonymous method can be converted into delegates with a variety of signatures. This is not possible with lambda expressions

Lambda expressions aid developers in efficiency in that they are a way to create methods in a far more productive manner than writing methods by hand. Anonymous methods allow you to hook an action directly to an event as opposed to having a separate event handler. Anonymous methods also let you define a delegate without having to create a method name.