**All the answers should be written inside Program.cs.**

* **Short-answer (theory) questions → should be written as comments (// or /\* ... \*/).**
* **oop questions → all classes should be placed in the same Program.cs**

**Questions**

1. Describe the problem generics address.
2. How would you create a list of strings, using the generic List class?
3. How many generic type parameters does the Dictionary class have?
4. True/False. When a generic class has multiple type parameters, they must all match.
5. What method is used to add items to a List object?
6. Name two methods that cause items to be removed from a List.
7. How do you indicate that a class has a generic type parameter?
8. True/False. Generic classes can only have one generic type parameter.
9. True/False. Generic type constraints limit what can be used for the generic type.
10. True/False. Constraints let you use the methods of the thing you are constraining to.

**Practice Exercise**

**Task 1**:  
 Define a generic class called **MyStack<T>** with the following requirements:

1. Use Stack<T> internally to store the data.
2. Implement a Count() method that returns the number of elements in the stack.
3. Implement a Pop() method that returns and removes the top element of the stack.
4. Implement a Push(T obj) method that adds an element to the stack.

Finally, create an instance of **MyStack<int>**, push two integers into it, and print out the current number of elements in the stack.

**Task 2**:  
 Create a generic repository pattern in C# with the following requirements:

1. Define a generic interface **IGenericRepository<T>** where **T : class**.  
     
   * The interface should declare the following methods:  
       
     + Add(T item)
     + Remove(T item)
     + Save()
     + IEnumerable<T> GetAll()
     + T GetById(int id)
2. Implement a class **GenericRepository<T>** that inherits from **IGenericRepository<T>**.  
     
   * Use a private List<T> field to store the data.
   * In the constructor, initialize the list as a new empty List<T>.
   * Provide method implementations for **Add**, **Remove, GetAll, GetById. No actual implementation is needed for Save.**