1. Implement a linked list. It should support the following operations:
   1. Check for empty.
   2. Add an element at the beginning.
   3. Add an element before a specified element.
   4. Add an element after a specified element.
   5. Delete a specified element.
   6. Print list contents.

Use templates to make the list a generic data type so that any data type can be stored in the list.

1. Write a program that asks for a number, validates that it is a prime number, and then prints a Pascal’s triangle with as many rows as the number. The program will be considered complete only when no arrays or vectors are used.
2. Implement a three level class hierarchy for 2D shapes. The base class (interface) must have methods for perimeter and area. Use these classes to write a program that asks the user for the number of shapes to be made. Make that number of shapes randomly. Iterate through the created shapes and output its type, size, perimeter and area. The program should demonstrate the following:
   1. Polymorphism through references.
   2. C++11 smart pointers.
   3. C++11 range for.
   4. Factory pattern.
   5. Abstract classes.
   6. Use of default, delete, override and final keywords.
3. Implement a base class Animal. It should have a method getAnimal() and getName(). The class should store the name of the animal as well. For derived concrete classes, calling these methods would return the type and name of the animal.

Implement an interface Speaker with the method speak(). For classes implementing this interface, calling this method would return the actual sound that concrete class makes.

Derive at least five actual animals from Animal. Some of them would also derive from Speaker. The main program would present a menu to the user to either create an animal from the given list or show all the created animals. In the first case, ask for the name of the animal as well. To show the animals, iterate through the created animals and print a greeting of the form: “<*sound*>! I am a/an <*animal*> named <*name*>”. Note that the <sound> clause would be absent for non-speakers. Examples:  
Woof! I am a dog, named Tommy.  
I am a rabbit named Fluffy. *🡨 Note that rabbit is not a speaker*  
Hiss! I am a snake, named Vasuki.  
To be considered complete, the program should demonstrate the following:

* 1. Polymorphism through references.
  2. C++11 smart pointers.
  3. C++11 range for.
  4. Multiple inheritance.
  5. Dynamic cast.
  6. Abstract classes.
  7. Use of default, delete, override and final keywords.