In Lab CA Revision Sheet – Not the same as the CA

1. Develop a **hierarchy of Employee classes**.
a) Employee is an ***abstract* class**. It has 2 subclasses, `Management` and `SalesStaff`.
The `Management` class has 4 data members, `empNum, name, department` and `salary`.
The `Sales` class also has 4 data members, `empNum, name, commission` and `salary`.

b) Shapes have a method `remuneration ()`which will calculate the salary of any Employee instance.
The remuneration of a Management object is equal to their salary attribute.
The remuneration of a SalesStaff object is salary + commission attributes
The remuneration method to use with a Shape pointer or reference will be determined based on the actual type to which the Shape points. (**i.e. Polymorphism**).

c) Employee has a method `ostream& display(ostream str)` which will display details of the Employee to the `ostream,` and then return this `ostream`.
This method will also be used **polymorphically**. Details are the type of the Employee, the values in the data members and the remuneration of the shape.

d) Provide an overloaded operator as a member function of the employee class, which will determine if one `Employee` is less than another based on their remuneration. The signature of this overloaded operator is: `bool operator< (const Employee &rhs);`
Note: this method will be implemented only in the Employee class, not in its subclasses.

e) Provide an overloaded operator as a **stand-alone function** to output the details of a `Employee`. The implementation will make use of the display method which is used **polymorphically.** The signature of the overloaded operator is
`ostream& operator<< (ostream &str, const Employee &employee);`

2. Write a template function that will take 3 arguments (passed by reference) and sort them so that the first argument holds the smallest item and the 3rd argument holds the largest item, and the second argument holds the middle item.

3. Create a template class which can be implemented as a linked list of any type. Test your class by declaring a linked list of doubles and a linked list of chars. Add some appropriate items to each list, and display the two lists.

4. Write a template class that implements a set of items. A set is a collection of items which does not allow duplicates. There should be no restriction on the size of the set. (to achieve this, I suggest holding a linked list of the items as a data member of the set. Use the linked list class you have developed in part 3)
The class should allow the user to:
a) Add a new item to the Set (note that the set should not allow duplicates)
b) Get the number of items in the set (this can be a data member of the Set)
c) Print out all the items in the Set.
Test your class by creating sets of different data types (e.g., integers, strings)