By now you should have a good understanding about why your program in the previous lab behaved like that. In this lab, you will try to modify your code to achieve polymorphism through virtual function and pointer/reference.

a. Make necessary changes to your class implementation
b. Keep the following main function as you had in your previous lab:

- In this main function, you first create an array of 3 elements of with type Employee.
- Now, define three new variable objects: the first variable is called emp0, the data type of it is Executive, the second variable is called emp1, the data type of it is Software Engineer, the fourth variable is called emp2, the data type of it Test Engineer. You should give the necessary information to create each object (i.e., needed by the constructor), for example, the following pass in the first name, last name, base salary, job title, bonus, profit sharing for an executive object.
  Executive emp0("John", "Doe", 100000, "VP", 1000000, 2000);

- Give the three array elements the value of the Executive object (emp0), the Software Engineer object (emp1) and the Test Engineer object (emp2) respectively. For example:
  emp[0] = emp0;
  etc.

- Call the Display Information method to display information for all three each employees using the following, for example,
  emp[0].DisplayInformation();
  etc.

- Record the output of your program here:

- Call the Display Information method to display information for all three employees using a different approach, for example,
  emp0.DisplayInformation();
  etc.

- Record the output of your program here:

- Could you explain why dynamic binding is still not achieved even though you have modified your member function to virtual functions?

c. Now add the following code to the existing main function above:

- Define a pointer of Employee type:
  Employee *empptr;

- Then make empptr pointing to an employ object and then call DisplayInformation()
  empptr = new Employee(… …);
  empptr->DisplayInformation();

- Then make empptr pointing to an Executive object and then call DisplayInformation()
  empptr = new Executive(… …);
  empptr->DisplayInformation();

- Then make empptr pointing to an Engineer object and then call DisplayInformation()
  empptr = new Engineer(… …);
  empptr->DisplayInformation();

- Record the output of your program here:

- Could you explain what kind of binding (static/dynamic) is used here?