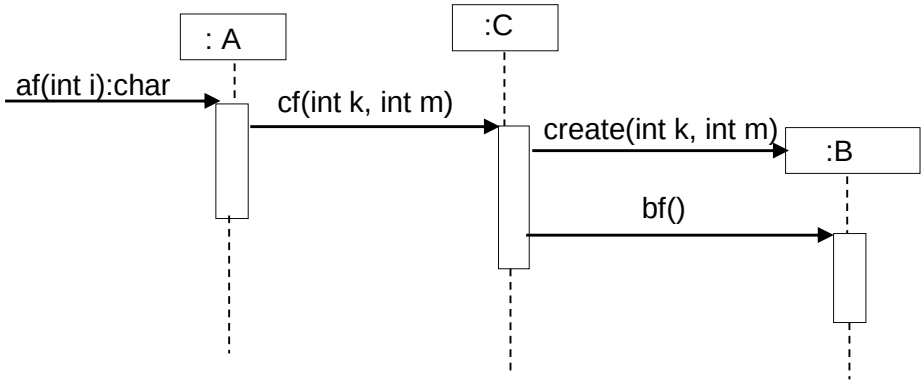
Object Oriented Modeling and Design Final Exam
Information about Solutions

QUESTION 1:

c)



Now A is coupled only to C, not to B.

QUESTION 2:

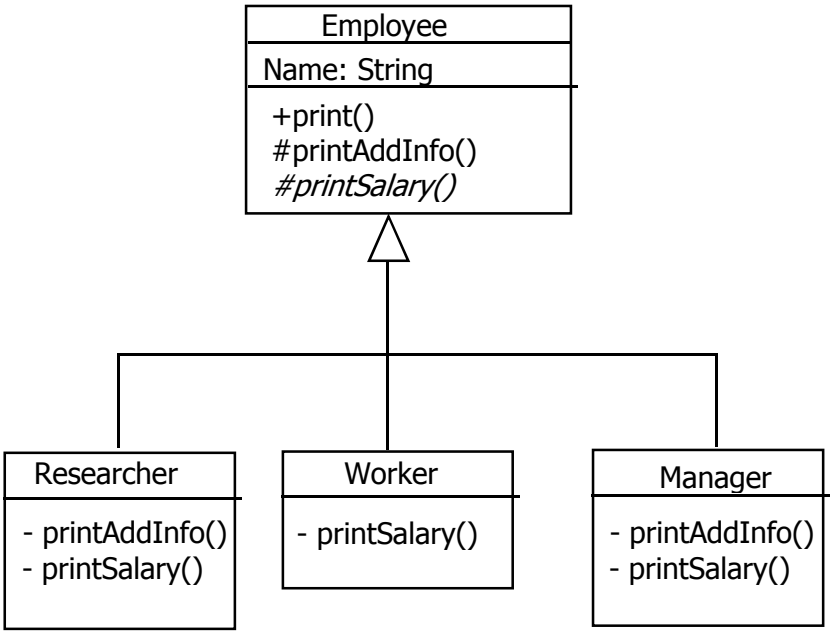
The information of different employees must be printed in a fixed format as follows.

Name: Common for all employees
Additional information: It varies and Workers don't have additional information.
Salary: All employees have salary but it is printed differently for different types.

a)

If you add a new employee you have to remember this format (steps).
You have to write everything again (code duplication).
If you put common functions into the base class you must remember to call these functions in the derived (child) classes. "Call-super" anti-pattern.

b) Solution is to apply the "Template Method" pattern.



```
class Employee{
   public:
     void print();  // Template Method

  protected:
     void printAddInfo(){};     // Default hook operation does nothing
     virtual void printSalary()=0;  //Virtual (abstract) primitive op.
  private:
     String Name;
}

void Employee::print            // Template method
{
   cout << Name << endl;
   printAddInfo();
   printSalary();
}
```

In all subclasses primitive methods (`printSalary()`) must be implemented. The hook method (`printAddInfo()`) will be implemented only if necessary (for example not in Worker).

QUESTION 3:

Necessary patterns: Bridge, Adapter and Singleton-Factory.
Employees (abstraction) are separated from the accounting programs (implementation). A bridge (pointer or reference) will connect these two groups. To create a common interface for different external accounting programs adapters will be used. The factory will create the proper adapter.

To add a new accounting program a new adapter must be written. Then the creation algorithm in the factory must be according to new program modified.