1. Code duplication is one type of “code smell”. If there is code duplication among several member functions within one class. What is the best solution for eliminating the smell?

a) creating a super-class for this class
b) creating a new member method of this class
c) redesign this class using the strategy design pattern

2. What type of refactoring can fix the code smell below?
   a) extracting more methods for Expression class
   b) changing parameter of the evaluation method
   c) extracting sub-classes from Expression class
   d) changing method names

```cpp
eenum opcode_t {ADD, SUB, MUL, DIV, NEG, CONST};
class Expression{
  private:
    Expression operand1;
    Expression operand2;
    opcode_t op;
    float const_value;
  public:
    Expression();
    float evaluation();
  ...
}
float Expression::evaluation()
{
  switch(op){
    case ADD: return operand1.evaluation() + operand2.evaluation();
    case SUB: return operand1.evaluation() - operand2.evaluation();
    case MUL: return operand1.evaluation() * operand2.evaluation();
    case DIV: return operand1.evaluation() / operand2.evaluation();
    case NEG: return - operand1.evaluation();
    case CONST: return const_value;
    case default: exit(1);
  }
}
```
3. Please refactor the code below to reduce code duplication. Please make changes to **both** the class diagram **and** the code below.

(see next page for the changed class diagram)

```cpp
float LaborItem::getUnitPrice()
{
    return emp.getRate();
}

float LaborItem::getTotalPrice()
{
    return emp.getRate() * quantity;
}

float NonLaborItem::getUnitPrice()
{
    return unitprice;
}

float NonLaborItem::getTotalPrice()
{
    return unitprice * quantity;
}
```

```cpp
+ float Item::getTotalPrice()
+     return getUnitPrice() * quantity;
+
```
Item

int quantity;
float getUnitPrice() = 0;
float getTotalPrice();

NonLaborItem

float unitprice;
float getUnitPrice();

LaborItem

Employee emp;
float getUnitPrice();

05/10/2016