CSCI 135 Software Design and Analysis, C++
Lab 7
Solution

Saad Mneimneh
Hunter College of CUNY

Lab A: PiggyBank
Consider the following class:

```
class PiggyBank {
  int a[4];
  //a[0] quarters
  //a[1] dimes
  //a[2] nickels
  //a[3] pennies

  bool breakable; //false by default
  bool emptied; //initially false,
                //set to true the first time piggy bank is emptied

 public:
  PiggyBank() {...}
  PiggyBank(int cents) {...}
  PiggyBank(int cents, bool breakable) {...}
  void add(int cents) {...}
  PiggyBank add(PiggyBank& b) {...} //by reference to empty b
  int empty() {...}
  void print() {...}
};
```

(a) Implement the default constructor to create an empty piggy bank, i.e. with 0 cents.

(b) Implement the second and third constructors to create a piggy bank with the specified amount of cents by first using as many quarters as possible, then as many dimes as possible, then as many nickels as possible, then pennies. We call this the fewest coins condition. In addition, the third constructor determines whether the piggy bank must be broken in order to get the money.

(c) Implement the add function to add the specified amount of cents to the piggy bank and rearrange to reestablish the fewest coins condition.

(d) Implement the empty function to return the number of cents contained in the piggy bank and reset the piggy bank to empty. If the piggy bank must be broken, the piggy bank cannot be used later, i.e. not matter what functions are

called it will always contain 0 cents (used emptied to keep track).

(e) Implement the second add function that takes another piggy bank as parameter and consolidate the two into a third one while reestablishing the fewest coins condition. In addition, the two piggy banks should be emptied (see above).

(f) Implement the print function to output information about the piggy bank, i.e. the number of quarters, the number of dimes, the number of nickels, and the number of pennies.
Solution:

```
#include <iostream>

using std::cout;

class PiggyBank {
  int a[4];
  bool breakable;
  bool emptied;

  void adjust(int cents) { //for convenience because it is repeated
    a[0]=cents/25;         //this is how we achieve the fewest coin condition
    cents=cents%25;        //it's private so it cannot be explicitly called
    a[1]=cents/10;
    cents=cents%10;
    a[2]=cents/5;
    a[3]=cents%5;
  }

 public:
  PiggyBank() {
    a[0]=a[1]=a[2]=a[3]=0; //assignment operator is right associative
    breakable=false;       //so a[3]=0 is done first, and returns 0
    emptied=false;
  }

  PiggyBank(int cents) {
    adjust(cents);
    breakable=false;
    emptied=false;
  }

  PiggyBank(int cents, bool breakable) {
    adjust(cents);
    this->breakable=breakable; //I am using the implicit pointer
    emptied=false;             //to avoid shadowing, can be solved
  }                            //by renaming the parameter

  void add(int cents) {
    if (!breakable || !emptied) //check if it can be reused
      adjust(a[0]*25+a[1]*10+a[2]*5+a[3]+cents);
  }
```

```
  PiggyBank add(PiggyBank& b) {  //without reference, we would only empty the copy
    return PiggyBank(empty()+b.empty());
  }

  int empty() {
    int cents=a[0]*25+a[1]*10+a[2]*5+a[3];
    a[0]=a[1]=a[2]=a[3]=0;
    emptied=true;
    return cents;
  }

  void print() {
    cout<<a[0]<<" quarters, "<<a[1]<<" dimes, "
        <<a[2]<<" nickels, "<<a[3]<<" pennies\n";
  }
};

int main() {
  PiggyBank b;
}
```