

This worksheet is for your use during and after lecture. It will not be collected or graded, but I think you will find it a useful tool as you learn C++ and study for the exams. Explain all false answers for the “True or False” questions; in general, show enough work and provide enough explanation so that this sheet is a useful pre-exam review. I will be happy to review your answers with you during office-hours, via Email, or instant messaging.

1. Use the following code listing to answer the questions beginning on page 1.

```
10  /**
11   * Wallet interface
12   */
13  class Wallet {
14  public:
15     // default ctor makes an empty, anonymous wallet
16     Wallet( );
17     Wallet( const string& theName );
18     Wallet( const string& theName,
19            int theDollars, int thePennies );
20     // retrieve the wallet identification
21     string id( ) const;
22     // retrieve the monetary value
23     double getValue( ) const;
24     // track expenses
25     void expense( int costDollars, int costPennies );
26     void expense( double costAmount );
27     // track income
28     void income( int earnDollars, int earnPennies );
29     void income( double earnAmount );
30     // read value from the input stream —
31     // returns a reference to the istream argument provided
32     istream& input( istream& is );
33     // print myself to the output stream
34     // returns a reference to the ostream argument provided
35     ostream& output( ostream& os ) const;
36 private:
37     // makes sure monetary amounts are always positive ,
38     // carries >= 100 pennies to dollars
39     // displays a message to cout if amount is negative
40     bool money_logic( int theDollars, int thePennies ) const;
41     // makes sure monetary amounts are positive
42     // displays a message to cout if amount is negative
43     bool money_logic( const double theAmount ) const;
44     // converts dollars and pennies to a decimal dollar amount
45     double dollars_pennies_to_value( int theDollars,
46                                     int thePennies ) const ;
47
48     // data members
49     string name; // owner name
50     double value; // amount of money in wallet, >=0
51 };
```

- (a) This is the class definition. True or false? Explain why your answer is correct.

- (b) Why don't the function prototypes on lines 16 and 17 have return types?
- (c) How many member functions does `Wallet` have? How many data members?
- (d) How many of `Wallet`'s member functions may change the calling object state?
- (e) Where will `name` and `value` be initialized with values?

2. Consider this implementation for `Wallet`'s three parameter constructor prototyped at line 19 on the preceding page.

```
3 void Wallet_3ctor( string& theName, double theCash, int theChange ) const
4 {
5     name = theName;
6     value = theCash + double(theChange)/100;
7 }
```

State six things wrong with the constructor implementation.

3. (a) Write the prototype for a `Wallet::get_dollars` function that returns the total value of bills in a `Wallet` object as an integer.

(b) Now write the implementation for part a.

4. (a) Write the prototype for a `Wallet::clone` function: it returns a new `Wallet` object that is a duplicate of the calling object, **except** with a parameter specified new name.

(b) Now write the implementation for part a.

- (c) Suppose a `main()` routine has created a `Wallet` object named `A`. Using the results of parts a and b, what C++ statement(s) can be added to `main` to create another `Wallet` object named `B` with the same amount of money as `A` but with a wallet name or id of `"CashFlow"`?

5. Suppose you have a C++ routine with an already created `Wallet` object named `aWallet` that was created with
- ```
Wallet aWallet("Jackson", 20, 20);
```

Use the implementation of `Wallet::output` shown below to answer parts a and b.

```
101 // print myself to the output stream
102 // returns a reference to the ostream argument provided
103 ostream& Wallet::output(ostream& os) const
104 {
105 os << name << "_wallet_contains_" << value;
106 return os;
107 }
```

- (a) Write **two** C++ statements that would print `aWallet` to `cout` followed by an `endl`.
- (b) Now, write the C++ statement (note the singular, **one statement**) that would print `aWallet` to `cout` followed by the phrase "is not enough money.", and terminated with `endl`.
- (c) Write the C++ statement that would read **the output** of part a into `aWallet` if it were typed at the keyboard or read from a file.

6. Suppose the merger function defined below belonged to the Wallet class.

```

24 /**
25 * Domestic "mergers" frequently result in the
26 * redistribution of funds as well as an agreed upon
27 * portion to be used as initial savings.
28 *
29 * In the end, money is simply moved around — not
30 * created or consumed.
31 */
32 Wallet Wallet::merger(Wallet& w)
33 {
34 // remember these values
35 double totalValue(value + w.value);
36
37 // the new investment Wallet
38 Wallet cookieJar("Investments");
39
40 // divvy up the amounts
41 value *= 0.5;
42 w.value *= 0.75;
43
44 // redistribute funds — no funds created or lost
45 cookieJar.value = value + w.value;
46 value = (totalValue - cookieJar.value) / 2;
47 w.value = (totalValue - cookieJar.value) / 2;
48
49 return cookieJar;
50 }
51
52 int main()
53 {
54 Wallet Alice("Alice", 80000, 0); // 79K
55 Wallet Bob("Bob", 1200, 0); // 1.2K
56 Wallet Savings = Alice.merger(Bob);
57
58 Savings.output(cout) << endl;
59 Alice.output(cout) << endl;
60 Bob.output(cout) << endl;
61 return 0;
62 }

```

(a) What will the main routine print as the value of each wallet after the merger() function is called?

(b) Would the result change if the collision had been Savings = Bob.merger( Alice)?  
Explain:

7. Consider the following true or false questions. You may simply identify a true statement as “true,” and move on to the next question. **Rewrite any false statements** into a true statement about the same fundamental idea.
- (a) The `private` segment of a class should be used to hide secret or confidential information.
  - (b) Constructor prototypes have a return type of `void`.
  - (c) Constructor prototypes may not have empty or `void` argument lists.
  - (d) The compiler must see the class declaration (or *interface*) after the class definition.
  - (e) The `public` class segment is used for publishing data members to web sites.
  - (f) A C++ *default constructor* accepts zero arguments.
  - (g) The compiler provides all classes with a default constructor.
  - (h) “Helper” functions are `public` or `private` member functions that encapsulate chunks of class logic for reuse by application developers.