**Exam #4 Sample Questions**

You are to write some overloaded operator methods for coinage objects that keep track of the amount of coins you have. A Coinage class will be used with the following six private data members:

integer **p** for the number of pennies (eg 6)

integer **n** for the number of nickels (eg 4)

integer **d** for the number of dimes (eg 3)

integer **q** for the number of quarters (eg 5)

integer **num** for the total number of coins (eg 18 (6 + 4 + 3 + 5))

float **amt** for the total money amount for the coinage (eg 1.81 (6 \* .01 + 4 \*.05 + 3 \* .10 + 5 \* .25))

(3) 1. Overload the <= operator in the public section. Example: if (c1 <= c2) Your code should return true if the total money amount for c1 is less than or equal to the total money amount for c2.

bool operator<= (const Coinage &rhs) const

{ return amt <= rhs.amt; }

(3) 2. Overload the + operator. Example: total = c1 + c2; Your code should add together the two Coinage objects c1 and c2 and place the total of these two objects into the total object. You will need to add together the corresponding data members and place these sums into total. For example, if c1 has 6 pennies and c2 has 10 pennies, then total has 6 + 10 =16 pennies.

Coinage Coinage::operator+( const Coinage &rhs ) const

{

return Coinagel(p + rhs.p, n + rhs.n, d + rhs.d, q + rhs.q, num + rhs.num, amt + rhs.amt);

}

(3) 3. Overload the ++ operator (preincrement). Example: ++c1; Your code should add one of each coin type to the c1 object. That is, c1 should have one additional penny, nickel, dime and quarter after the preincrement. Do not forget to recalculate the necessary data members.

Coinage &Coinage::operator++() // preincrement

{

p++; n++; d++; q++;

num += 4;

amt = (.01 \* p) + (.05 \* n) + (.1 \* d) + (.25 \* q);

return \*this;

}

(Note that four new coins are being added to the object so must do num += 4; instead of num++; or ++num;)

(3) 4. Overload the ++ operator (postincrement). Example: c1++; Your code should add one of each coin type to the c1 object. That is, c1 should have one additional penny, nickel, dime and quarter after the postincrement. Do not forget to recalculate the necessary data members.

Coinage Coinagel::operator++(int) // postincrement

{

Coinage temp = \*this;

p++; n++; d++; q++;

num += 4;

amt = (.01 \* p) + (.05 \* n) + (.1 \* d) + (.25 \* q);

return temp;

}

(4) 5. Overload the stream extraction operator. Example: cin >> c1;

istream &operator>>( istream &input, Coinage &c)

{

input >> c.p >> c.n >> c.d >> c.q;

c.num = c.p + c.n + c.d + c.q;

c.amt = (.01 \* c.p) + (.05 \*c.n) + (.1 \*c.d) + (.25 \* c.q);

return input;

}

(4) 6. Overload the stream insertion operator. Example: cout << c1;

ostream &operator<<( ostream &output, const Coinage &c)

{

output << c.p << c.n << c.d << c.q << c.num << c.amt << endl;

return output;

}

(for stream extraction, 3 steps: extract the values from the input stream that the user entered, calculate the necessary data members, and return the input stream.)