# Sample Midterm Exam Solution
# EECS 211
# Tuesday, February 11, 2014

NAME _____KEY_____

Problem 1. (10 points) Consider the following C++ function:

```
int f(int N)
{
   int i;

   if (N <= 0)
      return –1;
   i = 1;
   do
   {
      i  =  i + i;
   } while (i <= N);
   return i / 2;
}
```

a. What value is returned when this function is called with N = 10?

8

b. What value is returned when this function is called with N = -7?

-1

c. What value is returned when this function is called with N = 1?

1

Problem 2. (10 points) The following section of code is supposed to read an integer N from the user and print all the square integers from 1 up to and including N. The code compiles, but there are two logical errors. Tell what they are and how to fix them.

```
int j, N;
~~cin >> j;~~     cin >> N;
for (j = 1;  ~~j <= N~~;  j++)    j*j <= N (or using sqrt()…)
   cout << j*j << " is a square number.\n";
```

Problem 3. (20 points) An electronics parts warehouse that sells integrated circuits has the following pricing structure.

- Each part has a base price.
- If a customer buys 1 to 99 pieces, the price for each piece is the base price.
- If a customer buys 100 to 499 pieces, the price per piece is 65% of the base price.
- If a customer buys 500 to 999 pieces, the price per piece is 40% of the base price.
- If a customer buys 1000 or more pieces, the price per piece is 30% of the base price.

For example, if a customer orders a part which has base price $1.50 and orders 600 pieces, the charge for the parts would be 0.4 * 1.50 * 600. That is, the customer would be charged 40% of $1.50 for each piece, and the total for 600 pieces is 600 times 40% of $1.50.

There is also a shipping and handling charge computed as follows:

- For every order there is a flat handling charge of $7.50.
- There is a postage charge of $0.03 per piece ordered.

Write a function which takes two parameters – a double number giving the base price of a piece and an integer number giving the number of pieces ordered. The function returns the total cost of the order.

```
double cost(double base_price, int num_pieces)
{
   double discount, price, postage;

   if (num_pieces >= 1000)
      discount = 0.3;
   else if (num_pieces >= 500)
      discount = 0.4;
   else if (num_pieces >= 100)
      discount = 0.65;
   else
     discount = 1.0;

   price = base_price * num_pieces * discount;
   postage = 7.5 + 0.03 * num_pieces;
   return price + postage;
}
```

Make sure parameter, return, and variable types are reasonable (float/double vs. int)
Make sure local variables are declared properly
Make sure discount and postage are properly calculated and correct value returned
Possible: maybe using ints and report cost in terms of cents?

Problem 4. (25 points) Consider the following C++ class used to represent the liquid volume of various containers.

```
class LiquidVolume
{
private:
   double number_of_ounces;
public:
   LiquidVolume(double v);          // Constructor 1
   LiquidVolume();                  // Constructor 2
   void setVolume(double newv);
   double getVolume();
   double timesLarger(LiquidVolume* otherLV);
};
LiquidVolume::LiquidVolume(double v)          { number_of_ounces = v; }
LiquidVolume::LiquidVolume()                  { number_of_ounces = 0; }
void LiquidVolume::setVolume(double newv)     { number_of_ounces = newv; }
double LiquidVolume::getVolume()              {  return number_of_ounces; }
```

a. For each variable in the following declaration, tell which constructor gets called:
LiquidVolume v1, v2(24.0), v3(17.5), v4[10];

v1: Constructor 2
v2: Constructor 1
v3: Constructor 1
v4: Constructor 2

b. Given the above declaration, what is the effect of the following statement:
v1.setVolume(v3.getVolume()+6.0);

The volume of v1 is set to be 23.5 (6.0 more than the volume of v3).

c. Implement the timesLarger function. The argument otherLV represents another LiquidVolume object. The timesLarger function should calculate how many times larger this object's volume is than otherLV's volume, i.e., the ratio of this volume over otherVolume's volume. Return -1.0 if otherLV has volume 0.

```
double timesLarger(otherLV* other) {
   if (otherLV == NULL  || other->number_of_ounces == 0)  return -1.0;
   return number_of_ounces / other->number_of_ounces;
}  NULL check not necessary, though it is a good idea…
```

d. A LiquidVolume object is supposed to represent the physical volume of a container. The constructor and setVolume functions, while syntactically correct, have a logical flaw. What is it, and how should it be fixed?

The volume of a container typically does not change, so setVolume is logically inconsistent.

Problem 5. (35 points) Suppose you are a sports fan and want to write a program to keep track of your favorite sports teams. You decide you need a **class** called **Team** to hold information about an individual team.

a. List five data members you would have in this class.

```
char* name;
Player* member;
char* mascot;
int rank;
double rank_score;   //Anything remotely reasonable is okay
```

b. Describe 3 functions other than constructor/destructor you would provide.

```
char* getName();
void  setName(char* newname);
void updateRank(Team& opponent, bool win);
//Anything reasonable is okay
```

c. Write a C++ class declaration for your specification including an appropriate constructor. YOU DO NOT NEED TO IMPLEMENT YOUR MEMBER FUNCTIONS.

```
class Team
{
public:
   //Default constructor
   //OPTIONAL:  more constructors and destructor
   //Functions from above…
private:
  //Data members from above…
};  //Don't forget semicolon!
```

d. Of course you are a big fan and have many favorite sports teams. Write code to allocate **and** deallocate a catalog that can store up to myTeams teams, where myTeams is a previously defined int variable.

```
Team* catalog = new Team[myTeams];
delete[] catalog;
```