1. Consider the following complete program:

```
#include "std_lib_facilities.h"
void swap(double d1, double d2)
{    double temp = d1; // copy d1's value to temp
     d1 = d2;          // copy d2's value to d2
     d2 = temp;        // copy d1's old value to d2
}
int main()
{    double x = 1;
     double y = 2;
     cout << "x == " << x << " y== " << y << '\n';
     swap(x,y);
     cout << "x == " << x << " y== " << y << '\n';
}
```

   (a) What does the program print to the console?
   ANS:

   ```
   x == 1 y == 2
   x == 1 y == 2
   ```

   (b) How would you change it to actually do the intended swap?
   ANS: Pass the variables by reference, like so: `void swap(double& d1, double& d2)`

   (c) Describe how your change affects the program's use of variables and memory.
   Passing by reference creates an alias; that is, another name, for the object passed. The only memory used is the stack space allocated to hold the alias. The alias is essentially just a pointer to the object; that is, it contains the address in memory of the object to which it refers.

2. Consider the following complete program:

```
#include "std_lib_facilities.h"
class UDT {
public:
     int m;               // data member
     int mf(int v) { int old = m; m=v; return old; } // function member
};
int main()
{   UDT var;              // var is a variable of type UDT
     var.m = 7;           // assign to var's data member m
     cout << "\nm = " << var.m;
     int x = var.mf(9);   // call var's member function mf()
     cout << "\nm = " << var.m;
     return x;
}
```

   (a) What is printed to the console?
   ANS:

   ```
   m = 7
   m = 9
   ```

   (b) What value is returned by `main()` ?
   ANS: 7

   (c) How would you change the `UDT` to make the data member private?
   ANS: You'd need to proved `get()` and `set()` functions. See code at right:

```
class UDT {
private:
     int m;              // data member
public:
     void set_m(int x) {m = x; }
     int get_m() { return m; }
     int mf(int v) { int old = m; m=v; return old; }
};
int main()
{   UDT var;
     var.set_m(7);
     cout << "\nm = " << var.get_m();
     int x = var.mf(9);
     cout << "\nm = " << var.get_m();
     return x;
}
```

3. Consider the following complete program:

```
class T
{    int m;
     int mf() { return m; }
public:
     int f(int i) { m=i; return mf(); }
};
int main()
{    T t;
     int y = t.mf();
     //int y = t.f(8);
     return y;
}
```

(a) Explain in detail what the function `f()` does.
ANS: The public method `f()` of the `class T` takes an `int` and copies it to a memory location with the alias `i`, then assigns the value of `i` to the member variable of type `int`, named, `m` and calls the private member function `mf()` and returns its return value, `m`;

(b) As is, the program has an error. What is the error?
ANS: 'int T::mf()' on line 3 is private (in the context of the call on line 9).

(c) How would you modify the program so that it has no error and `main()` returns the value 8?
ANS: Delete or comment out line 9 and uncomment line 10. Voilá!

4. Consider program implementing a `Pet` class as partially listed below:

```
#include "std_lib_facilities.h"
class Pet {
private:
     string kind;
     string name;
     int birthday; // date in the form "20170101" for Jan 1, 2017
     int age; // pet's age
public:
     Pet(string k, string n, int bday) : kind(k), name(n), birthday(bday) { };
     Pet() { cout << "\nYou've created a Pet."; }
     string get_kind();
     string get_name();
     int get_age();
     void updateAge(int); /// use birthday
     void set_kind(string s) { kind = s; }
     void set_name(string n) { name = n; }
     void set_bday(int b) { birthday = b; }
     void speak();
};
void init_pet(Pet& p)
{    string k, n;
     int birth;
     cout << "\nWhat kind of pet are you creating? ";
     cin >> k;
     p.set_kind(k);
     /// Answer question 4b
}
int Pet::get_age()
{    /// answer question 4a
}
void Pet::speak()
{    if(kind=="dog")
          cout << "\nWoof! My name is "<<name<<" and I'm a "<<get_age()<<"-year old "<<kind;
     /// answer question 4c
}
void Pet::updateAge(int today) {
     /// answer question 4d
}
int main()
{    cout << "\nEnter today's date.  Use, say, 20161205 for Dec. 5, 2016: ";
```

```
    int today;
42  cin >> today;
    Pet dog("dog", "Bix", 20141231);
44  dog.updateAge(today);
    dog.speak();
46  Pet cat;
    init_pet(cat);
48  cat.updateAge(today);
    cat.speak();
50 }
```

That can produce this user interaction, when complete:

```
Enter today's date.  Use, say, "20001105" for "Nov. 5, 2000": 20161205

Woof! My name is Bix and I'm a 2-year old dog

You've created a Pet.
What kind of pet are you creating? cat
What is your pet's name? Fluffy
What is your pet's birthday? (use, say, "19990228" for "Feb. 28, 1999)": 10161205

Meow! My name is Fluffy and I'm a 1000-year old cat
```

(a) Provide an appropriate body for Pet's method `get_age()` ANS:

```
return age;
```

(b) Complete the definition of `init_pet()` so that line 47 will execute properly. ANS:

```
void init_pet(Pet& p)
{   string k, n;
    int birth;
    cout << "\nWhat kind of pet are you creating? ";
    cin >> k;
    p.set_kind(k);
    cout << "\nWhat is your pet's name? ";
    cin >> n;
    p.set_name(n);
    cout << "\nWhat is your pet's birthday? (use, say, \"19990228\" for \"Feb. 28, 1999)\": ";
    cin >> birth;
    p.set_bday(birth);
}
```

(c) Add code to `Pet`'s `speak()` method so that line 49 will execute properly. ANS:

```
 else if(kind=="cat")
        cout<<"\nMeow! My name is "<<name<<" and I'm a "<<get_age()<<"-year old "<<kind;
```

(d) Use integer division(/) and the modulo operator(%) so that `updateAge` gives the correct age. ANS:

```
void Pet::updateAge(int today) {
    if(birthday/100 % 100 >= today/100 % 100)
        age =  today/10000 - birthday/10000;
    else age = today/10000 - birthday/10000 - 1;
}
```

(e) Why is the parameter passed by reference in `init_pet()`?// ANS: If it were passed by value then when the local copy went out of scope it would be destroyed. For the value to persist in the calling function it needs to passed by reference.

5. Consider the following code fragment for the user-defined type, `Token`:

```
#include "std_lib_facilities.h"
class Token {
public:
    char kind;          // what kind of token
    double value;       // for numbers: a value
    Token(char ch)      // make a Token from a char
        :kind(ch), value(0) { }
    Token(char ch, double val)   // make a Token from a char and a double
        :kind(ch), value(val) { }
};

Token get_token();      // read a token from cin

vector<Token> tok;      // we'll put the tokens here

int main()
{   while (tok.size()<5) {
        Token t = get_token();
        tok.push_back(t);
    }
    double d = tok[0].value, temp;
    for (int i = 0; i<tok.size(); ++i) {
        temp = d;
        if (tok[i].kind=='*') {     // we found a multiply!
            d *= tok[i+1].value;
            cout << temp << '*' << tok[i+1].value << '=' << d << endl;
        }
    }
}
Token get_token() {
    char ch;
    double val;
    cin >> ch;
    switch(ch) {
    case '2': case '3': {
        cin.putback(ch);
        cin >> val;
        return Token('8',val);
    }
    case '*':
        return Token(ch,0);
    }
}
```

(a) What type of object is the variable `tok` in this program and what is its scope?
ANS: The variable `tok` is a `vector<Token>` with global scope.

(b) Describe what happens on lines 17-20 of `main()`. What, specifically is produced if the user enters "3*2*3"?
ANS: The variable `tok` is initially empty so `tok.size()` is zero and we enter the `while` loop and assign the `token ('8',3)` to the variable `t` which is then pushed onto the `tok`, making its size = 1. This processed is looped until `tok` contains the `tokens` {`('8',3),('*',0),('8',2),('*',0),('8',3)`} at which point `tok.size()==5` and the `while` loop is exited.

(c) Make a table threading the values of `i`, `d`, and `temp` for the code in lines 21-28 if the user enters "3*2*3".

i	d	temp
0	3	3
1	6	3
2	18	6
3	18	6
4	18	18

(d) What is output to the console if the user enters "3*2*3"?

```
3*2*3
3*2=6
6*3=18
```

(e) How would you modify the code so the the result of entering "1*2/2*2" is computed correctly according to the order of operations? (ie, left to right for multiplication and division). You will need to modify both `main()` and `get_token()`.

ANS: Modify `get_token()` to include the `'1'` case and the division case:

```
Token get_token() {
    char ch;
    double val;
    cin >> ch;
    switch(ch) {
    case '1': case '2': case '3': {
        cin.putback(ch);
        cin >> val;
        return Token('8',val);
    }
    case '*':
        return Token(ch,0);
    case '/':
        return Token(ch,0);
    }
}
```

Also, modify `main()` to include this condition in the `for` loop:

```
        if (tok[i].kind=='/') {    // we found a division!
            d /= tok[i+1].value;
            cout << temp << '/' << tok[i+1].value << '=' << d << endl;
        }
```