1. Consider the flow chart for applying the bubble-sort algorithm for sorting objects that can be put in order.

   (a) Tabulate values for the `counter`, `i` and the current state of `vector<char> v={ a,l,g,o,r,i,t,h,m}`.

i	counter	v
1	1	{ a,l,g,o,r,i,t,h,m}
⋮	⋮	⋮

ANS: Ooops! This flow chart was cobbled from something on the internets that purported to be bubble, but, upon close inspection is not! Implementing the flow chart algorithm (as shown below) we have

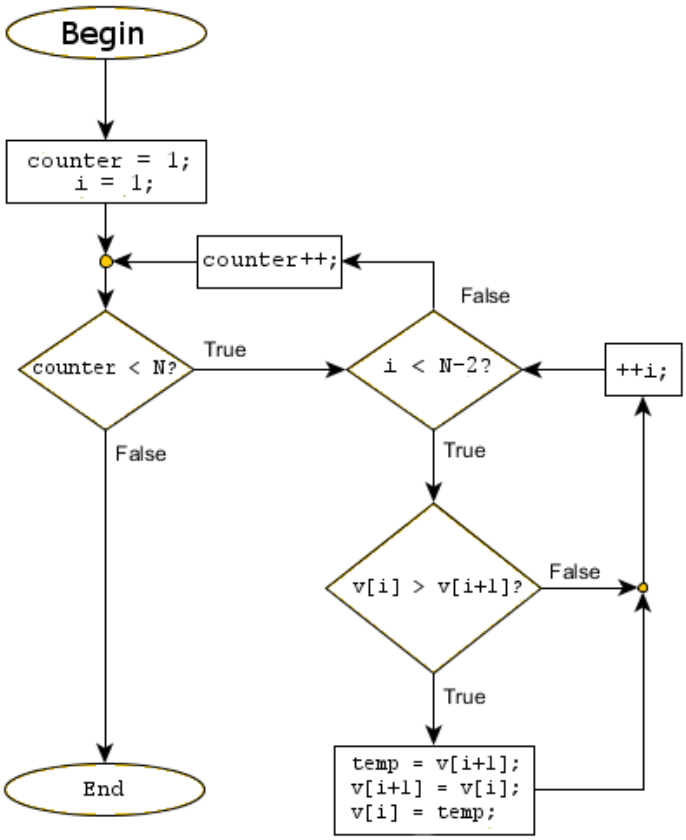
```
1  void print(vector<char> v) {
       for(int i = 0; i < v.size(); ++i)
3          cout << v[i];
       cout << endl;
5  }

7  int main() {
       //int counter{1};
9      vector<char> v{'a','l','g','o','r','i','t','h','m'};
       int counter=1, i = 1;
11     while(counter < v.size()) {
           for(;i < v.size()-2; i++) {
13             if(v[i]>v[i+1]) swap(v[i],v[i+1]);
               cout << "i = " << i << ", ctr = " << counter << ", ";
15             print(v);
           }
17         ++counter;
           cout << "\ncounter = " << counter;
19     }
   }
```

m

which will produce this self-documenting output:

```
i = 1, ctr = 1, aglorithm
i = 2, ctr = 1, aglorithm
i = 3, ctr = 1, aglorithm
i = 4, ctr = 1, agloirthm
i = 5, ctr = 1, agloirthm
i = 6, ctr = 1, agloirhtm
counter = 2
counter = 3
counter = 4
counter = 5
counter = 6
counter = 7
counter = 8
counter = 9
```



(b) Write a C++ function that implements the algorithm.
The algorithm of the flow chart is shown above, but doesn't successfully sort vector of characters. A proper bubble-sort algorithm involves loops as shown below:

```
void bsort(vector<char>& v) {
    for (int i=1; i < v.size(); i++)
        for (int j=1; j <= v.size()-i; j++) {
            if(v[j-1] > v[j]) swap(v[j-1], v[j]);
            cout << "i = " << i << ", j = " << j << ", ";
            print(v);
        }
// INVARIANT: v[n-i], ..., v[n-1] are in their correct positions
}
```

which produces this self-tabulating output:

```
i = 1, j = 1, algorithm                 i = 3, j = 4, agilohmrt
i = 1, j = 2, aglorithm                 i = 3, j = 5, agilhomrt
i = 1, j = 3, aglorithm                 i = 3, j = 6, agilhmort
i = 1, j = 4, aglorithm                 i = 4, j = 1, agilhmort
i = 1, j = 5, agloirthm                 i = 4, j = 2, agilhmort
i = 1, j = 6, agloirthm                 i = 4, j = 3, agilhmort
i = 1, j = 7, agloirhtm                 i = 4, j = 4, agihlmort
i = 1, j = 8, agloirhmt                 i = 4, j = 5, agihlmort
i = 2, j = 1, agloirhmt                 i = 5, j = 1, agihlmort
i = 2, j = 2, agloirhmt                 i = 5, j = 2, agihlmort
i = 2, j = 3, agloirhmt                 i = 5, j = 3, aghilmort
i = 2, j = 4, agliorhmt                 i = 5, j = 4, aghilmort
i = 2, j = 5, agliorhmt                 i = 6, j = 1, aghilmort
i = 2, j = 6, agliohrmt                 i = 6, j = 2, aghilmort
i = 2, j = 7, agliohmrt                 i = 6, j = 3, aghilmort
i = 3, j = 1, agliohmrt                 i = 7, j = 1, aghilmort
i = 3, j = 2, agliohmrt                 i = 7, j = 2, aghilmort
i = 3, j = 3, agilohmrt                 i = 8, j = 1, aghilmort
```

(c) How many times is the comparison `v[i] > v [i+1]` made in sorting { `a,l,g,o,r,i,t,h,m`} ?
ANS: In the given problem, this would be 6 comparisons. In a proper bubble sort, it'd be 9*8/2 = 36 comparisons.

2. Explain how and why the following program does what it does:

```
int main() {
    int a = 10;
    while(a<11) {
        int a = 12;
        cout << "\na = " << a;
        cin.get();
    }
}
```

ANS: What's happening here is there are two variables with the same name, `a`, but different scopes. The scope of the first `a=10` includes the conditional `while` for looping, while the scope of the second `a` is the body of the `while` loop. In the body of the loop `a` is set to 12, this value is printed to the console (for confirmation) and then the loop pauses to get input from the keyboard. However, this value of `a` is not used for the conditional, so it's an infinite loop.

3. Write a `class` named `Commodity` with data fields `string id; double unitPrice, inStock, sold;` and a member functions that include a constructor and a function `revenue()` that computes and returns the `unitPrice*sold`. In `main()` write a loop that allows the user to enter various `Commodities` into a `vector` of `Commodities` like, say, bananas, indicating the `id`, the `unitPrice`, the amount `inStock`, and the amount `sold`. Then compute the total revenue from all those `commoditys` and the total value `inStock`.

ANS: Here's some code (with parts borrowed from Daniel King).
First the header file, `commodity.h`

```
/// Definition for Commodity class

class Commodity{
public:
    Commodity(string, double, double, double);
    ///calculates the worth of the units in stock
    double storedMoney();
    ///calculates the worth of the units sold
    double revenue();
    double getSold() {return sold;}
    string getId() {return id;}
private:
    string id;
    double unitPrice, inStock, sold;
};
double Commodity::revenue() {
    return unitPrice*sold;
}
Commodity::Commodity(string ID, double UP, double IS, double S):
    id(ID), unitPrice(UP), inStock(IS), sold(S) {}
```

...and then the helper function and driver:

```
/// CS7A Midterm2 Spring 15 - problem 3
#include "std_lib_facilities.h"
#include "commodity.h"
void getCommodities(vector<Commodity>& vC) {
    char tempChar;
    string tempStr;
    double tempUnitPrice, tempInStock, tempSold;
    bool goodInput{true};
    cout << "Enter each Commodity as a \n"
         << "(1) id string (say \"bananas\)"\n"
         << "(2) unit price in dollars\n"
         << "(3) amount in stock, and\n"
         << "(4) amount sold (followed by an 'x' to exit):\n";
    while(goodInput){
        cin >> tempChar;
        ///checks if the input is x
        if(tempChar != 'x'){ ///if it isn't x, put it back
            cin.putback(tempChar);
        } else{ ///exit the program
            return;
        }
        goodInput = cin >> tempStr;
        ///if there's a dollar sign, this filters it out
        cin >> tempChar;
        if(tempChar != '$'){
            cin.putback(tempChar);
        }
        cin >> tempUnitPrice >> tempInStock >> tempSold;
        vC.push_back(Commodity(tempStr, tempUnitPrice, tempInStock, tempSold));
    }
}
int main() {
    vector<Commodity> vC;
    getCommodities(vC);
    double revenues{0};
    for(int i = 0; i < vC.size();++i) revenues += vC[i].revenue();
    cout << "\nSold\n";
    for(int i = 0; i < vC.size();++i) {
        cout << vC[i].getSold() << " " << vC[i].getId() << endl;
    }
    cout << "\nfor a total revenue of $" << revenues;
}
```

A typical run might look like this:

```
Enter each Commodity as a
(1) id string (say "bananas")
(2) unit price in dollars
(3) amount in stock, and
(4) amount sold (followed by an 'x' to exit):
bananas $1.20 100 50
strawberries $3.40 40 12
eggplant $0.80 34 20
x

Sold
50 bananas
12 strawberries
20 eggplant

for a total revenue of $116.8
```

4. Write a program that reads every other word of a text into a new file whose filename is supplied by the user. Execute this on the poem at http://geofhagopian.net/CS007A/cs7A-s15/scooping.txt into a new file. Submit both the file and its text file output on printout paper.

5. Consider the code below:

```
const char name = 'a'; // name token
const char let = 'L'; // declaration token
const string declkey = "let"; // declaration keyword
Token Token_stream::get() {
  if (full) {
    full = false;
    return buffer;
  }
  char ch;
  cin >> ch;
  switch (ch) {
  case quit: case print: case '(': case ')': case '+':
  case '-': case '*': case '/': case '%':
  return Token{ch}; // let each character represent itself
  case '.': // a floating-point-literal can start with a dot
  case '0': case '1': case '2': case '3': case '4':
  case '5': case '6': case '7': case '8': case '9': // numeric literal
  { cin.putback(ch); // put digit back into the input stream
    double val;
    cin >> val; // read a floating-point number
    return Token{'8',val}; // let '8' represent ``a number"
  }
  if (isalpha(ch)) {
    cin.putback(ch);
    string s;
    cin>>s;
    if (s == declkey) return Token(let); // declaration keyword
    return Token{name,s};
  }
  default:
    error("Bad token");
  }
}
```

(a) What is the name of the function defined here?

(b) What is the scope of this function?

(c) The function handles input from the console. Describe, in step by step detail, how it would handle the console input

`"let GC = 6.67384e-11"`