```
//Class: CSE 330
// Term: Spring 2014
// Instructor: George M. Georgiou
// Name: Seth Lemanek
// Homework 1
// Title: Vector.h
#ifndef VECTOR_H
#define VECTOR_H
#include <algorithm>
#include <assert.h>
using namespace std;

template <class T> class vector {
public:
    typedef T * iterator;
    typedef T value_type;
      //constructors
    vector  () { buffer = 0; resize(0); }
    vector  (unsigned int size) { buffer = 0; resize(size); }
    vector  (unsigned int size, T initial);
    vector  (vector & v);
    ~vector  () { delete [ ] buffer; }

     //member functions
    T      back () { return buffer [mySize -1];}
    iterator      begin() { return buffer; }
    int           capacity () { return myCapacity; }
    bool   empty () {return mySize == 0; }
    iterator      end () { return begin() + mySize; }
    T      front () { return buffer[0]; }
    void   pop_back () { mySize --; }
    void   push_back (T value);
    void   reserve (unsigned int newCapacity);
    void   resize (unsigned int newSize) { reserve(newSize); mySize = newSize; }
    int           size () { return mySize; }

    //operator
    T & operator [ ] (unsigned int index) { return buffer[index]; }

protected:
    unsigned int mySize;
    unsigned int myCapacity;
    T * buffer;
};

template <class T> vector<T>::vector (unsigned int size, T initial)
//creates vector with given size
//initialize all elements with given parameter
{
      buffer = 0;
      resize(size);
      fill (begin(), end(), initial);
}

template <class T> vector<T>::vector (vector & v)
//creates vector by copying from a previous one
{
      buffer = 0;
      resize(v.size());
      copy (v.begin(), v.end(), begin());
}

template <class T> void vector<T>::reserve (unsigned int newCapacity)
//reserves new capacity as large as the argument given
{
      if (buffer == 0)
        {
         mySize = 0;
         myCapacity = 0;
         }
      if (newCapacity <= myCapacity)
         return;
      T * newBuffer = new T [newCapacity];
      assert (newBuffer);
      copy (buffer, buffer + mySize, newBuffer);
      myCapacity = newCapacity;
      delete [ ] buffer;
      buffer = newBuffer;
}

template <class T> void vector<T>::push_back (T value)
//pushes a value to the back of the vector
{
      if (mySize >= myCapacity)
          reserve(myCapacity + 5);
      buffer [mySize++] = value;
}

#endif
```

```
//Class: CSE330
//TErm: Spring 2014
////Instructor: George M. Georgiou
//Name: Seth Lemanek
//Homework 1
//Title: Vector Assignment
#include<iostream>
#include<fstream>
#include<math.h>
#include<assert.h>
#include"vector.h"

using namespace std;

template<class T>
T average (vector<T> values, T size)
{
 if (size != 0)
 {
    T total = size;
    T sum = 0;
    for (int i = 0; i < size; i++)
      sum += values[i];//continues to add up until size is reached
    T result = sum/total;
    return result;
 }
 else//the size is zero
    return 0;
}

template<class T>
T variance(vector<T> values, T size)//finds variance
{
    T total = size;
    T sum = 0;
    T diff = 0;
    T avg = average(values, size);
    for(int i = 0; i < size; i++)
    {
        diff = (values[i] - avg)*(values[i] - avg);//sum((a(i)-avg)^2)
      sum += diff;
    }
    T result = sum/(total - 1);
    return result;
}
int main()
{
    vector<float> nums (100);
    float num;
    int i = 0;
    cout << "Do you want the program to read from a file? (y/n) \n";
    char response;
    cin >> response;
    if(response == 'y')//user must type in name of file
    {
      string name;
      cout << "Please enter the name of the file: \n";
      cin >> name;
      ifstream infile;
      infile.open(name.c_str());
            if(cin.fail())
      {
          cout << "Error: failure to find file!\n";
      }
        while (infile.good())
      {
          infile >> num;
          nums[i] = num;
          i++;
      }
      infile.close();
    }
    else//user types in the set of numbers then type a letter
    {
      cout << "Please type the numbers then type a character.\n";

            while (cin.good())
      {
         if (i < nums.size())
         {
         cin >> num;
         nums[i] = num;
```

```
          }
          else//if counter i indicates number goes over capacity
          nums.resize(1);
          i++;
        }
      }
      float n = 0.0;
      for (int j = 0; j < nums.size(); j++)
      {
        if(nums[j] != 0)
          n++;//count all numbers not zero
      }
      cout << nums[0] << nums[1] << nums[2] << endl;
      cout << "Average: " <<  average(nums, n) << endl;
      cout << "Variance: " << variance(nums, n) << endl;

}
```

```
Script started on Wed 30 Apr 2014 11:07:02 AM PDT
#]0;004470530@jb358-27:/students/csci/004470530/cse330/HW1##[?
1034h[004470530@jb358-27 HW1]$ ./a.out#######vi
numcounter.cpp#################[10P./a.out
Do you want the program to read from a file? (y/n)
y
Please enter the name of the file:
nums.txt
1025
Average: 8.625
Variance: 41.125
#]0;004470530@jb358-27:/students/csci/004470530/cse330/HW1#[004470530@jb358-27
HW1]$ ./a.out
Do you want the program to read from a file? (y/n)
n
Please type the numbers then type a character.
2
7
80
55
33# #4
6
78^[[A# ## ## ## #
n
2780
Average: 37.4286
Variance: 1157.95
#]0;004470530@jb358-27:/students/csci/004470530/cse330/HW1#[004470530@jb358-27
HW1]$ ./a.out#######vi numcounter.cpp#################[1@g+
+#[C#[C#[C#[C#[C#[C#[C#[C#[C#[C#[C#[C#[C#[C#[C###################[1Pvi#[C#[C#[C#[C#
[C#[C#[C#[C#[C#[C#[C#[C#[C#[C#[C##################[1@g+
+#[C#[C#[C#[C#[C#[C#[C#[C#[C#[C#[C#[C#[C#[C#[C###################[7Pvi
vector.h#######numcounter.cpp##[K##[K##[K##[K##[K##[K##[K##[K##[K##[K##[K##[K##[K#
#[K##[K##[K##[Kexit

Script done on Wed 30 Apr 2014 11:08:23 AM PDT
```