Homework #1 of 2

Write a program that reads in an elapsed time in hours, minutes, and seconds and converts it all into seconds. Use a structure called Time to hold the time: hours, minutes, and seconds.

New Homework #2

Find the prime numbers between 1 and N. N=100

Procedure:

List all numbers from 1 to N

1 is prime

2 is prime, but multiples of 2 are not prime, so mark all multiples of 2 until N

next unmarked number in list is 3, mark all multiples of 3 until N

find next unmarked number and continue

….

stop when next unmarked number^2 is greater than N. Why? *

Keep a list of array of values[N] with each element 0 or 1 if marked.

* Because all primes below this number say K can not be multiplied by this number as they would have been knocked out as nonprime earlier. So K(K+1) is next possible number.

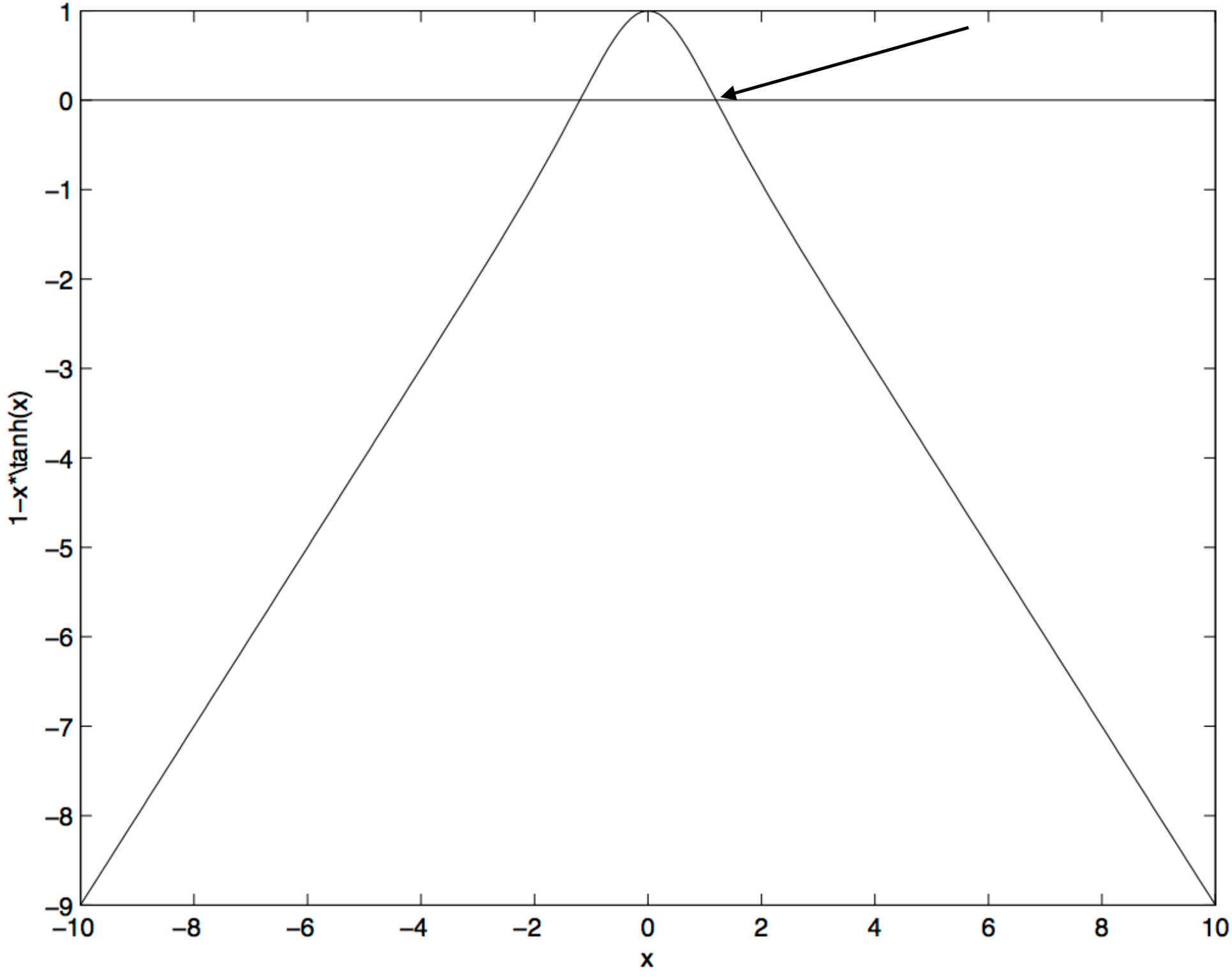
```
/*  Homework 1
    Find root of f(x) = a - x * tanh (x) by Newton's method
    tanh'(x) = 1/(cosh(x)*cosh(x))
    R.A. Dalrymple, Feb 2, 2014
    g++ Newton.cpp -o Newton
    */
#include <cmath>
#include <iostream>
#include <cstdlib>
using namespace std;

double NR_method (double a, double x)
 {
   double f = a - x*tanh(x);
   double fp =-tanh(x) - x/(cosh(x)*cosh(x));
   return x-f/fp;
   }

int main(int argc, char *argv[])
{
   cout<<"Usage: Newton a, x0:" <<endl;
   double a = atof(argv[1]);
   double x = atof(argv[2]);
   for (int i = 0 ; i<3 ; i++)
     {
       x = NR_method(a, x);
       cout <<"x = " <<  x <<", f(x) = " << a- x*tanh(x) << endl;
       }
   return 0;
}
```

Plot of 1- x*tanh(x)

The solution: ±1.199678



Arrays and Matrices

Single dimension a[10]
Two dimensions b[10][10]
M dimensions c [5][5]…[5]

Declare arrays at the beginning

```
float a[10];                //allocates space for 10 floats

int b[5] = {1, 2, 3, 4, 5};
int b[5] = {};      //initializes to zeros

 for (int i=0; i< 4;i++)
    cout << scientific <<b[i]<<endl;
```

Static arrays: dimension known at compile time

Dynamic Arrays

Sometimes you don't know the size of array at compile time

```
int * ptr;                  //ptr is a pointer pointing to an integer
ptr = new int ;          // allocates an array of no size;
or
ptr = new int [num];  //allocate size num
```

```
delete [] ptr;              //free up the memory of ptr when done
```

In C: these are equivalent to malloc and free

Allocate array

```
// rememb-o-matic
#include <iostream>
#include <new>                  //dynamic arrays library
using namespace std;
int main ()
{
  int i,n;
  int * p;
  cout << "How many numbers would you like to type? ";
  cin >> i;
  p= new int[i];                        //  <— allocate array
    for (n=0; n<i; n++)
    {
      cout << "Enter number: ";
      cin >> p[n];
    }
    cout << "You have entered: ";
    for (n=0; n<i; n++)
      cout << p[n] << ", “;
    cout <<endl;
    delete[] p;                         //    <— delete array!
    return 0;
}
```

Reading Files

We are familiar with reading and writing from terminal with cin and cout

- **`ofstream`**: Stream class to write on files
- **`ifstream`**: Stream class to read from files
- **`fstream`**: Stream class to both read and write from/to files.

```
// basic file operations
#include <iostream>
#include <fstream>
using namespace std;

int main () {
  ofstream myfile;
  myfile.open ("example.txt");
  myfile << "Writing this to a file.\n";
  myfile.close();
  return 0;
}
```

Reading data from file

file: data.txt

```
234.33  56.000  235656.1  52.222254555
21.2   15.6   17.23566
26.6
```

```
 //  dataRead.cc
//  Created by Munan Xu on 2014-02-04.
//
//  A simple program to read in whitespace separated floats from a text file
//
#include <iostream>
#include <fstream>
#include <iomanip>   //allow setprecision to get all the decimal points
using namespace std;
int main() {
        ifstream f("data.txt");
        double dataArray[1000];
        int numNum = 0;
        while (!f.eof()) {
                f >> dataArray[numNum++] ;
        }
        for(int i = 0; i < numNum-1; i++)
                cout << setprecision(10) << dataArray[i] << endl; //needs <iomanip>
               //cout << scientific <<dataArray[i]<<endl;
              //cout << fixed <<dataArray[i]<<endl;
}
```

Vectors

Vectors are arrays, but dynamically allocated
(means that their size can be changed)

```
/* examples with vector
*/

#include <iostream>
#include <vector>

using namespace std;

int main (int argc, char* argv [])
{
  vector<int> vectorOne(10,5);   // vectorOne has 10 elements, initialized with 5
  vector<int> vectorTwo(10);   // 10 elements, initialized as zero
  vector<int> VectorUnknown;  //initialized, but empty—dynamic allocation
  vector<float> test(5,1.0);  //only 5 elements (given the value 1.0)
  for (int i = 0; i<10; i++)
    cout << "i: " <<vectorOne[i] <<"," <<vectorTwo[i] <<"," <<test[i] <<endl;
   }
```

Vector functions

vector<int> vectorOne(10,5); //given

VectorOne.at(i); same as VectorOne[i]; but better

VectorOne.size();

VectorOne.resize(num)
VectorOne.resize(num, value);   //note the overload
appends value or can truncate vector

Inserting into Vector

To insert an element at end of vector (instead of resize)

push_back(value);
e.g. VectorOne.push_back(5); //5 added to end

To remove an element from end:

pop_back();
VectorOne.pop_back();

Adding/removing from arbitrary location:

insert(location, value);
erase(location);
erase(start, end); where start/end are *iterators*

Iterators

```
// vector::begin/end
#include <iostream>
#include <vector>

int main ()
{
  std::vector<int> myvector;
  for (int i=1; i<=5; i++) myvector.push_back(i);

  std::cout << "myvector contains:";
  for (std::vector<int>::iterator it = myvector.begin() ; it != myvector.end(); ++it)
    std::cout << ' ' << *it;
  std::cout << '\n';

  return 0;
}
```

Multi-dimensional Vectors

Declaration:

vector<vector<float>> newVector;

A vector of vectors:

Think of a column vector with each element being a row vector

Change dataRead.cc to fill a vector

Using vector<double>dataVector;

dataReadVector.cc

```
//  dataReadVector.cc
//
//  A simple program to read in a 1D vector
//
#include <iostream>
#include <fstream>
#include <iomanip>   //allow setprecision to get all the decimal points
#include <vector>
using namespace std;
int main() {
double temp;
   ifstream f("data.txt");
   vector<double> dataVector;
   int numNum = 0;
   while (!f.eof()) {
         f >> temp;
         dataVector.push_back(temp) ;
         ++numNum;
   }
   for(int i = 0; i < numNum-1; i++)
         cout << setprecision(10) << dataVector.at(i) << endl; //needs <iomanip>
   }
```

Reading strings

```
std::vector<std::string> lines;
for (std::string line; std::getline( ifs, line ); /**/ )
   lines.push_back( line );
```

ifs is the file name

```
//  Vector2D.cc
//
//  A simple program to make a two-dimensional array
//
#include <iostream>
#include <fstream>
#include <vector>
using namespace std;
int main() {
      vector<vector<double> > dataVector;  //initialize with no size
      for (int i = 0; i < 50; i++) {
     vector<double> row; // Create an empty row
       for (int j = 0; j < 10; j++) {
         row.push_back(float(i * j)); // Add an element to the row equal to i*j
       }
    dataVector.push_back(row); // Add the row to the main vector
}
       for(int i = 0; i < 50; i++)
         {
          for (int j=0; j<10; j++)
              {cout << dataVector[i][j]<<", ";   //here's how to address each element
              }
         cout <<endl;
         }
}
```

```
//  Vector2DRead.cc
//
//  A simple program to read a two-dimensional array of floats from file
//
#include <iostream>
#include <fstream>
#include <iomanip>   //allow setprecision to get all the decimal points
#include <vector>
#include <string>
using namespace std;
int main() {
        ifstream f("data2.txt");
        vector<vector<double> > dataVector;  //initialize with no size
        for (int i = 0; i < 4; i++)
        {
      vector<double> row; // Create an empty row
       for (int j = 0; j < 2; j++)   //for the row
       {
         double temp;
         f>> temp;
         row.push_back(temp); // Add an element to the row
       }
       dataVector.push_back(row); // Add the row to the main vector
      }
         for(int i = 0; i < 4; i++)
           {
            for (int j=0; j<2; j++)
                  {cout <<setprecision(10) ,, dataVector[i][j]<<", ";   //here's how to address each element
                  }
           cout <<endl;
           }
}
```