Istvan von Fedak

```
/*
 * main.cpp
 *
 *  Created on: Sep 12, 2016
 *      Author: Istvan
 */
#include "Person.h"
#include <iostream>
#include <fstream>

int determineSize(string s){
	/*
	 * Determins the size of the array based on the imput file.
	 */
	ifstream fin(s.c_str());
	if(!fin){
		cout<< "couldn't open the file \n";
		return 0;
	}
	else{
		int lineCount=0;
		string line;
		//counting lines
		while(getline(fin,line) && !fin.fail()){
			lineCount++;
		}
		return lineCount;
	}
}
void print(Person p[], const int n){
	for(int i = 0; i < n;i++){
		cout<<i<<" ";
		p[i].print();
	}
}
void finData(string s, Person p[]){
	/*
	 * It populates the array of books.
	 */
	ifstream fin(s.c_str());
	if(!fin){
```

```
            cout<< "couldn't open the file \n";
      }
      else{
            //Populating the array with information.
            string ssn,fn,ln,dob;
            for(int i = 0; !fin.fail(); i++){
                  fin>>ssn;
                  if(fin.fail()) break;
                  fin>> fn;
                  if(fin.fail()) break;
                  fin>>ln;
                  if(fin.fail()) break;
                  fin>>dob;
                  if(fin.fail()) break;
                  p[i].setPerson(ssn,fn,ln,dob);
            }
      }
}

void bubbleSort(Person a[], const int n){
      Person temp;
      for(int i = 0; i< n; i++){
            for(int j = 0; j<(n-1)-i; j++){
                  if(a[j].age()>a[j+1].age()){
                        temp = a[j+1];
                        a[j+1]= a[j];
                        a[j]=temp;
                  }
            }
      }
}
void shakerSort(Person a[], const int n){
      Person temp;
      for(int i = 0; i< n; i++){
            //left -> right
            for(int j = i; j<(n-1)-i; j++){
                  if(a[j].age()>a[j+1].age()){
                        temp = a[j+1];
                        a[j+1]= a[j];
                        a[j]=temp;
                  }
            }
            //right -> left
```

```
            for(int j = (n-1)-i; j>i; j--){
                  if(a[j].age()<a[j-1].age()){
                        temp = a[j-1];
                        a[j-1]= a[j];
                        a[j]=temp;
                  }
            }
      }
}
void selectionSort(Person a[], const int n){
      int maxIndex;
      Person temp;
      for(int i = n-1; i>0; i--){
            //find max index
            maxIndex =i;
            for(int j = 0; j <i;j++){
                  if(a[maxIndex].age()<a[j].age()) maxIndex = j;
            }
            //swap
            if(a[maxIndex].age()!=a[i].age()){
                  temp = a[maxIndex];
                  a[maxIndex] = a[i];
                  a[i]=temp;
            }
      }
}
void insertionSort(Person a[], const int n){
      Person tmp;
      for (int i = 1; i < n; i++) {
            for (int j  = i;a[j - 1].age() > a[j].age() && j > 0; j--)
{
                  tmp = a[j];
                  a[j] = a[j - 1];
                  a[j - 1] = tmp;
            }
      }

 }

void reverse(Person p[], const int n){
      /*
       * I could pass a boolean function into the sorting functions,
       * but I found this easier to debug and doesn't take a long time.
```

```
	 * I would have to change the comparisons on the sorting
functions
	 * to change the conditions for the sorting algorithms. Or I can
	 * pass a boolean parameter that tells the function if it's
	 * ascending(true) or decending(false) with and if(ascending)
	 * sort assending, else sort decending, but personally I din't
	 * like the way the long functions looked.
	 */
	Person temp;
	for(int i = 0; i < n/2; i++){
		temp = p[i];
		p[i] = p[(n-i)-1];
		p[(n-i)-1] = temp;
	}
}
int main(){

	cout<<"Running...\n\n";
	/*
	 * Change the imput file here to whatever file you want to read
into the person array
	 * as long as it's in the format ssn fn ln YYYYMMDD\n.
	 */
	string s = "database1.txt";

	//This function finds the number of lines the fin has.
	int n = determineSize(s);

	// After determining the size, the array of books is created.

	Person *p = new Person[1000];
	finData(s, p);
	/*
	//I commented out this section because of automation, if you want
to check my
	//sorting algorithms just uncomment this section.
	cout<<"Populated the array with the file \n";
	finData(s, p);

	bubbleSort(p,n);
	shakerSort(p,n);
	selectionSort(p,n);
	insertionSort(p,n);
```

```
	print(p,n);
	//*/

	/*AUTOMATION SECTION
	 * This section is to measure the time it takes to sort an array.
	 * It starts by defining the fin file name ex: s =
"database20.txt";
	 * It then deletes the array stored in the heap and creats one
according
	 * to the size of the new file
	 * There are 4 sections, one for each sorting algorithms, that
each have a
	 * for loop. Each time the for loop restarts it fins the file to
	 * hace a clean array that repeat each sorting algorithm 3 times
to obtain an
	 * average. Each time it goes through the loop it prints out the
three
	 * different times it took to sort, sort the sorted array, and
sort
	 * the reversed array.
	 *
	 * You can change the database by modifying the string s
	 */

	// <- delete one "/" to comment this whole section out

	//initialize doubles to measure time and index.
	double beforeUnsorted, afterUnsorted,beforeSorted, afterSorted,
beforeReversed, afterReversed,i;

	////////////////////////////////////////////////////////////////
/////
	s = "database1.txt";
	delete[] p; //prevents leaks
	n = determineSize(s);
	p = new Person[n];
	cout<<" Using bubble sort for "<<s<<endl;
	for(i =1; i<=3;i++){
		finData(s, p);

		beforeUnsorted = getCPUTime();
		bubbleSort(p,n);
```

```
            afterUnsorted = getCPUTime();

            beforeSorted = getCPUTime();
            bubbleSort(p,n);
            afterSorted = getCPUTime();

            reverse(p,n);

            beforeReversed = getCPUTime();
            bubbleSort(p,n);
            afterReversed = getCPUTime();
            cout<<"\n ================================== \n";
            cout<<" Trial number: "<< i<<endl;
            cout<< "Unsorted time = "<<afterUnsorted-
beforeUnsorted<<endl;
            cout<< "Sorted time = "<<afterSorted-beforeSorted<<endl;
            cout<< "Reversed time = "<< afterReversed -
beforeReversed<<endl;
      }

      cout<<"\n ////////////////////////////////////////////";
      cout<<"\n ////////////////////////////////////////////\n";

      cout<<" Using Shaker sort for:"<<s<<endl;
      for(i =1; i<=3;i++){
            finData(s, p);

            beforeUnsorted = getCPUTime();
            shakerSort(p,n);
            afterUnsorted = getCPUTime();

            beforeSorted = getCPUTime();
            shakerSort(p,n);
            afterSorted = getCPUTime();

            reverse(p,n);

            beforeReversed = getCPUTime();
            shakerSort(p,n);
            afterReversed = getCPUTime();
            cout<<"\n ================================== \n";
            cout<<" Trial number: "<< i<<endl;
```

```
            cout<< "Unsorted time = "<<afterUnsorted-
beforeUnsorted<<endl;
            cout<< "Sorted time = "<<afterSorted-beforeSorted<<endl;
            cout<< "Reversed time = "<< afterReversed -
beforeReversed<<endl;
      }
      cout<<"\n ////////////////////////////////////////////////";
      cout<<"\n ////////////////////////////////////////////////\n";
      cout<<" Using selection sort for:"<<s<<endl;
      for(i =1; i<=3;i++){
            finData(s, p);

            beforeUnsorted = getCPUTime();
            selectionSort(p,n);
            afterUnsorted = getCPUTime();

            beforeSorted = getCPUTime();
            selectionSort(p,n);
            afterSorted = getCPUTime();

            reverse(p,n);

            beforeReversed = getCPUTime();
            selectionSort(p,n);
            afterReversed = getCPUTime();
            cout<<"\n ================================== \n";
            cout<<" Trial number: "<< i<<endl;
            cout<< "Unsorted time = "<<afterUnsorted-
beforeUnsorted<<endl;
            cout<< "Sorted time = "<<afterSorted-beforeSorted<<endl;
            cout<< "Reversed time = "<< afterReversed -
beforeReversed<<endl;
      }

      cout<<"\n ////////////////////////////////////////////////";
      cout<<"\n ////////////////////////////////////////////////\n";

      cout<<" Using insertion sort for:"<<s<<endl;
      for(i =1; i<=3;i++){
            finData(s, p);

            beforeUnsorted = getCPUTime();
            insertionSort(p,n);
```

```
            afterUnsorted = getCPUTime();

            beforeSorted = getCPUTime();
            insertionSort(p,n);
            afterSorted = getCPUTime();

            reverse(p,n);

            beforeReversed = getCPUTime();
            insertionSort(p,n);
            afterReversed = getCPUTime();
            cout<<"\n ================================== \n";
            cout<<" Trial number: "<< i<<endl;
            cout<< "Unsorted time = "<<afterUnsorted-
beforeUnsorted<<endl;
            cout<< "Sorted time = "<<afterSorted-beforeSorted<<endl;
            cout<< "Reversed time = "<< afterReversed -
beforeReversed<<endl;
      }

cout<<"\n###############################################################
\n";
      ////////////////////////////////////////////////////////////////
////////
      ////////////////////////////////////////////////////////////////
////////
      //*/
      return 0;
}

/*
 * Date.h
 *
 *  Created on: Sep 12, 2016
 *      Author: Istvan
 */

#ifndef DATE_H_
#define DATE_H_

#include <iostream>
//#include <iomanip>
#include <string>
```

```
#include "support.h"

using namespace std;

class Date {
private:
      int year, month, day;

public:
      //////////code used for lab2///////////
      Date();
      int getDate();
      void setDate(int);
      void setDate(string YYYYMMDD);
      int age();
      ////////////////////////////////////////

      ////////////// More functions /////////
      /*
      Date(int YYYYMMDD);
      virtual ~Date();
      void setDate(int YYYYMMDD);



      int getYear(){return year;}
      int getMonth(){return month;};
      int getDay(){return day;}

      void print();
      void printNice();
      //*/
};


#endif /* DATE_H_ */

/*
 * Date.cpp
 *
 *  Created on: Sep 12, 2016
 *      Author: Istvan
 */
```

```cpp
//#include <iomanip>
#include "Date.h"

//////////Functions used for lab2////////////////////////////////
Date::Date() {
	year = 0;
	month = 0;
	day = 0;
}
int Date::getDate(){
	return ((year *pow(10,4)) + (month *pow(10,2)) + day);
}

void Date::setDate(int d){
	year = d/10000;
	month = (d/100)%100;
	day = d%100;
}

void Date::setDate(string d){
	int temp = string2int(d);
	year = temp/10000;
	month = (temp/100)%100;
	day = temp%100;
}

int Date::age(){
	return getCalendarDate()/10000 - year;
}

////////////////////////////////////////////////////////////

////////////// More functions ////////////////////////////////
/*
Date::Date(int d) {
	year = d/10000;
	month = (d/100)%100;
	day = d%100;
}
Date::~Date() {

}
```

```
void Date::setDate(int d){
	year = d/10000;
	month = (d/100)%100;
	day = d%100;
}



void Date::print(){
	cout << setfill('0') << setw(4)<< year;
	cout << setfill('0') << setw(2) << month;
	cout << setfill('0') << setw(2) << day;
}
void Date::printNice(){
	cout<<"Day: "<< setfill('0') << setw(2) << day;
	cout << ", Month: " << setfill('0') << setw(2)<< month;
	cout << ", Year: " << setfill('0') << setw(4) << year << endl;
}
//*/



/*
 * Person.h
 *
 *  Created on: Aug 25, 2016
 *      Author: Istvan
 */

#ifndef PERSON_H_
#define PERSON_H_

#include "Date.h"
using namespace std;

class Person{
private:
	int ssn;
	string firstName;
	string lastName;
	Date birthDate;
	int howOld;
```

```
public:
      //////////Functions used in lab2////////
      Person();
      void print();
      void setPerson(string ssn, string firstName, string lastName,
string birthDate);
      int age(){return howOld;}

      //If you want to do the comparisons without using howOld (the
long way)
      //comment out the age function above and uncomment the one below
      //int age(){return birthDate.age();}
      ////////////////////////////////////////

      ////////////// More functions //////////
      // one thats pretty cool is setPerson(string s)
      /*


      Person(int ssn, string firstName, string lastName, int
birthDate);
      Person(int ssn, string firstName, string lastName, Date
birthDate);
      Person(string ssn, string firstName, string lastName, string
birthDate);
      Person(string ssn, string firstName, string lastName, Date
birthDate);
      Person(int ssn, string fullName, int birthDate);
      Person(int ssn, string fullName, Date birthDate);
      Person(string ssn, string fullName, string birthDate);
      Person(string ssn, string fullName, Date birthDate);
      Person(string s);//sets the person from one string structured in
ssn firstName lastName birthDay


      void printNice();

      void setPerson(int ssn, string firstName, string lastName, int
birthDate);
      void setPerson(int ssn, string firstName, string lastName, Date
birthDate);
```

```
	void setPerson(string ssn, string firstName, string lastName,
Date birthDate);
	void setPerson(int ssn, string fullName, int birthDate);
	void setPerson(int ssn, string fullName, Date birthDate);
	void setPerson(string ssn, string fullName, string birthDate);
	void setPerson(string ssn, string fullName, Date birthDate);
	void setPerson(string s);//sets the person from one string
structured in ssn firstName lastName birthDay

	int getSnn(){return ssn;}
	string getHuman();
	string getFirstName(){return firstName;}
	string getLastName(){return lastName;}
	Date getBirthDate(){ return birthDate;}
	//*/


};

#endif /* PERSON_H_ */


/*
 * Person.cpp
 *
 *  Created on: Aug 25, 2016
 *      Author: Istvan
 */

#include "Person.h"
//////////Functions used in lab2////////
Person::Person(){
	ssn = 0;
	firstName =  "N/A";
	lastName = "N/A";
	birthDate.setDate(0);
	howOld = 0;
}
void Person::print(){
	cout<< ssn <<" "<< firstName << " " << lastName << " " <<
birthDate.getDate() << endl;
}
```

```
void Person::setPerson(string soNum, string fn, string ln, string
bDay){
      ssn = string2int(soNum);
      firstName =  fn;
      lastName = ln;
      birthDate.setDate(bDay);
      howOld = (getCalendarDate()/10000) - (string2int(bDay)/10000);
}
//////////////////////////////////////
////////////// More functions /////////
// one thats pretty cool is setPerson(string s)
/*

Person::Person(int soNum, string fn, string ln, int bDay){

      ssn = soNum;
      firstName =  fn;
      lastName = ln;
      birthDate.setDate(bDay);
      howOld = 0;
}
Person::Person(int soNum, string fn, string ln, Date bDay){

      ssn = soNum;
      firstName =  fn;
      lastName = ln;
      birthDate = bDay;
      howOld = 0;
}
Person::Person(string soNum, string fn, string ln, string bDay){
      ssn = string2int(soNum);
      firstName =  fn;
      lastName = ln;
      birthDate.setDate(bDay);
      howOld = 0;
}
Person::Person(string soNum, string fn, string ln, Date bDay){
      ssn = string2int(soNum);
      firstName =  fn;
      lastName = ln;
      birthDate = bDay;
      howOld = 0;
}
```

```
Person::Person(int soNum,string s, int bDay){
	ssn = soNum;
	birthDate.setDate(bDay);
	howOld = 0;
	//splitting the name;
	int lastSpaceIndex; // this int defines the index of the " " that
separates the last name from the first name.
	for(int i = s.length(); 0<=i;i--){
		if(s[i] == ' '){//starts at the last index of the full name
string and stops when it find the last " "
			lastSpaceIndex = i;//redefines the index of where the
" " is.
			while(i < s.length()){//records the last name
				lastName += s[i+1];
				i++;
			}
			for(int j = 0; j < lastSpaceIndex; j++){//records the
first name
				firstName += s[j];
			}
			break;
		}
	}
}
Person::Person(int soNum,string s, Date bDay){
	ssn = soNum;
	birthDate = bDay;
	howOld = 0;
	//splitting the name;
	int lastSpaceIndex; // this int defines the index of the " " that
separates the last name from the first name.
	for(int i = s.length(); 0<=i;i--){
		if(s[i] == ' '){//starts at the last index of the full name
string and stops when it find the last " "
			lastSpaceIndex = i;//redefines the index of where the
" " is.
			while(i < s.length()){//records the last name
				lastName += s[i+1];
				i++;
			}
			for(int j = 0; j < lastSpaceIndex; j++){//records the
first name
				firstName += s[j];
```

```
                }
                break;
            }
        }
}
Person::Person(string soNum,string s, string bDay){
        ssn = string2int(soNum);
        birthDate.setDate(bDay);
        howOld = 0;
        //splitting the name
        int lastSpaceIndex; // this int defines the index of the " " that
separates the last name from the first name.
        for(int i = s.length(); 0<=i;i--){
            if(s[i] == ' '){//starts at the last index of the full name
string and stops when it find the last " "
                lastSpaceIndex = i;//redefines the index of where the
" " is.
                while(i < s.length()){//records the last name
                    lastName += s[i+1];
                    i++;
                }
                for(int j = 0; j < lastSpaceIndex; j++){//records the
first name
                    firstName += s[j];
                }
                break;
            }
        }
}
Person::Person(string soNum,string s, Date bDay){
        ssn = string2int(soNum);
        birthDate = bDay;
        howOld = 0;
        //splitting the name
        int lastSpaceIndex; // this int defines the index of the " " that
separates the last name from the first name.
        for(int i = s.length(); 0<=i;i--){
            if(s[i] == ' '){//starts at the last index of the full name
string and stops when it find the last " "
                lastSpaceIndex = i;//redefines the index of where the
" " is.
                while(i < s.length()){//records the last name
                    lastName += s[i+1];
```

```
                    i++;
                }
                for(int j = 0; j < lastSpaceIndex; j++){//records the
first name
                    firstName += s[j];
                }
                break;
            }
        }
}
Person::Person(string s){
      //This function splits a string in the format ssn firstName
lastName YYYYMMDD
      //and stores it into the class Person.

      //local temporary variables
      howOld = 0;
      string ssNum,lName, fName, dob;

      bool sNum = true, fn = true, ln = true;
      for(int i = 0; i < s.length(); i++){
            if(sNum){
                  if(s[i] == ' ') sNum = false;
                  else ssNum += s[i];
            }
            else if(fn){
                  if(s[i] == ' ') fn = false;
                  else fName += s[i];
            }
            else if(ln){
                  if(s[i] == ' ') ln = false;
                  else lName += s[i];
            }
            else{
                  dob += s[i];
            }
      }
      ssn = string2int(ssNum), firstName = fName, lastName = lName,
birthDate.setDate(dob);
}


void Person::printNice(){
```

```
      cout<< ssn <<" "<< lastName << ", " << firstName << " " <<
birthDate.getDate() << endl;
}

void Person::setPerson(int soNum, string fn, string ln, int bDay){

      ssn = soNum;
      firstName =  fn;
      lastName = ln;
      birthDate.setDate(bDay);
}
void Person::setPerson(int soNum, string fn, string ln, Date bDay){

      ssn = soNum;
      firstName =  fn;
      lastName = ln;
      birthDate = bDay;
}

void Person::setPerson(string soNum, string fn, string ln, Date bDay){
      ssn = string2int(soNum);
      firstName =  fn;
      lastName = ln;
      birthDate = bDay;
}
void Person::setPerson(int soNum,string s, int bDay){
      ssn = soNum;
      birthDate.setDate(bDay);
      //splitting the name;
      int lastSpaceIndex; // this int defines the index of the " " that
separates the last name from the first name.
      for(int i = s.length(); 0<=i;i--){
            if(s[i] == ' '){//starts at the last index of the full name
string and stops when it find the last " "
                  lastSpaceIndex = i;//redefines the index of where the
" " is.
                  while(i < s.length()){//records the last name
                        lastName += s[i+1];
                        i++;
                  }
                  for(int j = 0; j < lastSpaceIndex; j++){//records the
first name
                        firstName += s[j];
```

```
                }
                break;
            }
        }
}
void Person::setPerson(int soNum,string s, Date bDay){
        ssn = soNum;
        birthDate = bDay;
        //splitting the name;
        int lastSpaceIndex; // this int defines the index of the " " that
separates the last name from the first name.
        for(int i = s.length(); 0<=i;i--){
            if(s[i] == ' '){//starts at the last index of the full name
string and stops when it find the last " "
                lastSpaceIndex = i;//redefines the index of where the
" " is.
                while(i < s.length()){//records the last name
                    lastName += s[i+1];
                    i++;
                }
                for(int j = 0; j < lastSpaceIndex; j++){//records the
first name
                    firstName += s[j];
                }
                break;
            }
        }
}
void Person::setPerson(string soNum,string s, string bDay){
        ssn = string2int(soNum);
        birthDate.setDate(bDay);
        //splitting the name
        int lastSpaceIndex; // this int defines the index of the " " that
separates the last name from the first name.
        for(int i = s.length(); 0<=i;i--){
            if(s[i] == ' '){//starts at the last index of the full name
string and stops when it find the last " "
                lastSpaceIndex = i;//redefines the index of where the
" " is.
                while(i < s.length()){//records the last name
                    lastName += s[i+1];
                    i++;
                }
```

```
                for(int j = 0; j < lastSpaceIndex; j++){//records the
first name
                     firstName += s[j];
                }
                break;
           }
     }
}
void Person::setPerson(string soNum,string s, Date bDay){
     ssn = string2int(soNum);
     birthDate = bDay;
     //splitting the name
     int lastSpaceIndex; // this int defines the index of the " " that
separates the last name from the first name.
     for(int i = s.length(); 0<=i;i--){
          if(s[i] == ' '){//starts at the last index of the full name
string and stops when it find the last " "
                lastSpaceIndex = i;//redefines the index of where the
" " is.
                while(i < s.length()){//records the last name
                     lastName += s[i+1];
                     i++;
                }
                for(int j = 0; j < lastSpaceIndex; j++){//records the
first name
                     firstName += s[j];
                }
                break;
           }
     }
}
void Person::setPerson(string s){
     //This function splits a string in the format ssn firstName
lastName YYYYMMDD
     //and stores it into the class Person.

     //local temporary variables
     string ssNum,lName, fName, dob;

     bool sNum = true, fn = true, ln = true;
     for(int i = 0; i < s.length(); i++){
          if(sNum){
                if(s[i] == ' ') sNum = false;
```

```
                else ssNum += s[i];
            }
            else if(fn){
                if(s[i] == ' ') fn = false;
                else fName += s[i];
            }
            else if(ln){
                if(s[i] == ' ') ln = false;
                else lName += s[i];
            }
            else{
                dob += s[i];
            }
      }
      ssn = string2int(ssNum), firstName = fName, lastName = lName,
birthDate.setDate(dob);
}

string Person::getHuman(){
      return firstName +" "+ lastName;
}
//*/


/*
 * support.h
 *
 *  Created on: Sep 6, 2016
 *      Author: nigel
 */

#ifndef SUPPORT_H_
#define SUPPORT_H_

#include <stdlib.h>
#include <string>
#include <ctime>


using namespace std;

//////////////////My functions//////////////////////
int getCalendarDate(void);
```

```
int string2int(string);
int pow(int, int);
////////////////////////////////////////////////

#if defined(__unix__) || (defined(__APPLE__) && defined(__MACH__))
#include <sys/time.h>
#include <sys/resource.h>
#elif defined(_WIN32)
#include <time.h>
#endif

double getCPUTime(void);
int randomInRange(const int start, const int end);

#endif /* SUPPORT_H_ */


/*
 * support.cpp
 *
 *  Created on: Sep 6, 2016
 *      Author: nigel
 */

#include "support.h"

//////////////////My functions////////////////////
int pow(const int m,const int n){
      if(n>0){
            int temp = m;
            for(int i = 1; i < n; i++){
                  temp *= m;
            }
            return temp;
      }
      else return 1;
}
int getCalendarDate(void){
      time_t t = time(0);   // get time now
      struct tm * now = localtime(&t);
      return ((now->tm_year + 1900)*10000 + (now->tm_mon + 1)*100 +
(now->tm_mday));
      delete now;
```

```
}
int string2int(string s){
      int i = 0, j = s.length() -1, temp,result = 0;
            char c;
            while(i < s.length()){
                  c = s[i];
                  temp = (int)c -48;
                  result += temp*pow(10,j);
                  i++;
                  j--;
            }
      return result;
}
////////////////////////////////////////////////

double getCPUTime(void) {
#if defined(__unix__) || (defined(__APPLE__) && defined(__MACH__))
      struct timeval tv;
      struct rusage ru;
      getrusage(RUSAGE_SELF, &ru);
      tv = ru.ru_utime;
      double t = (double)tv.tv_sec + (double)tv.tv_usec/1000000.0;
      tv = ru.ru_stime;
      t += (double)tv.tv_sec + (double)tv.tv_usec/1000000.0;
      return t;
#elif defined(_WIN32)
      return clock()/(double)CLOCKS_PER_SEC;
#endif
}


int randomInRange(const int start, const int end) {
#if defined(__unix__) || (defined(__APPLE__) && defined(__MACH__))
      return (start + random()%(end-start+1));
#elif defined(_WIN32)
      return (start + rand()%(end-start+1));
#endif
}
```