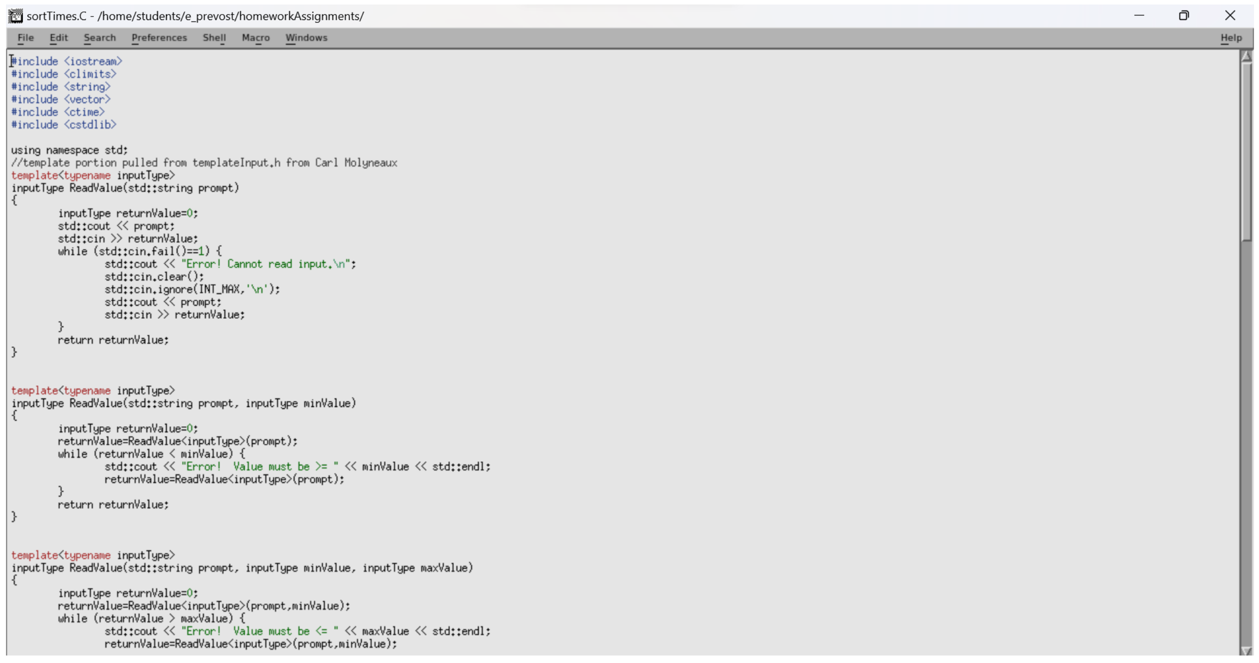
Sort Times- Robert P.

Test Outputs:

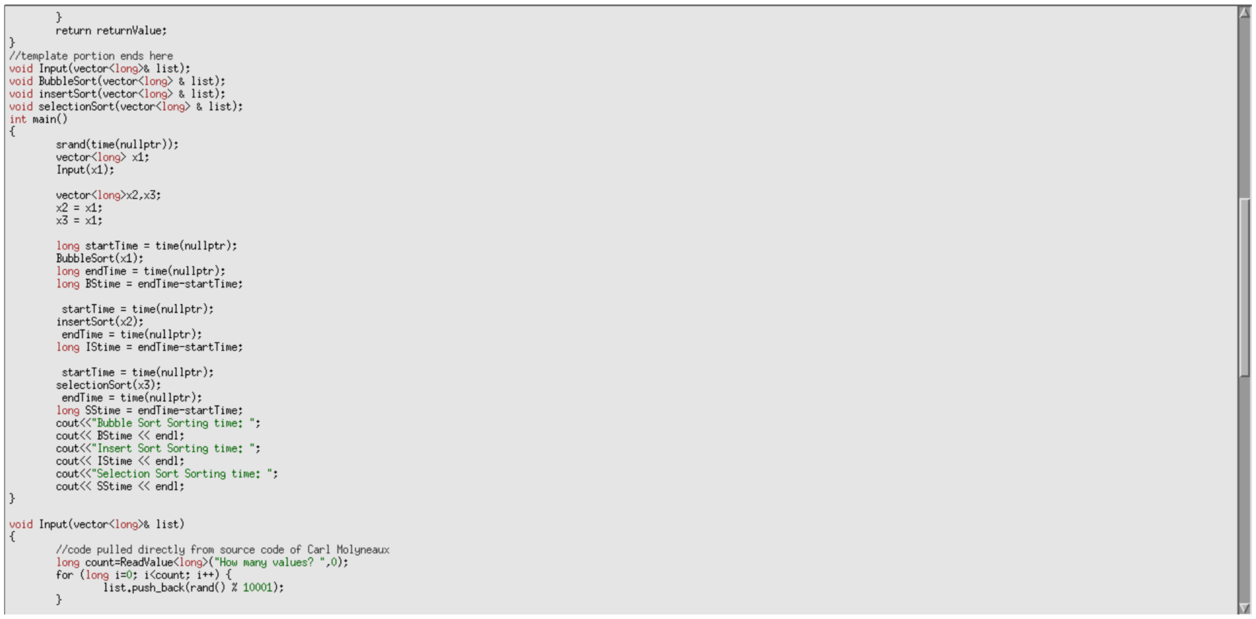
Vector Size	Bubble Sort Time (s)	Insert Sort Time (s)	Selection Sort Time (s)
1000	0	0	0
5000	1	0	0
10000	1	0	0
20000	4	1	2
50000	26	5	10

My observation is that Bubble Sort is the slowest by a significant margin and selection sort is the second slowest and insert sort is the third slowest. It looks like the times it takes for insert sort is directly half of the time it takes for selection sort.

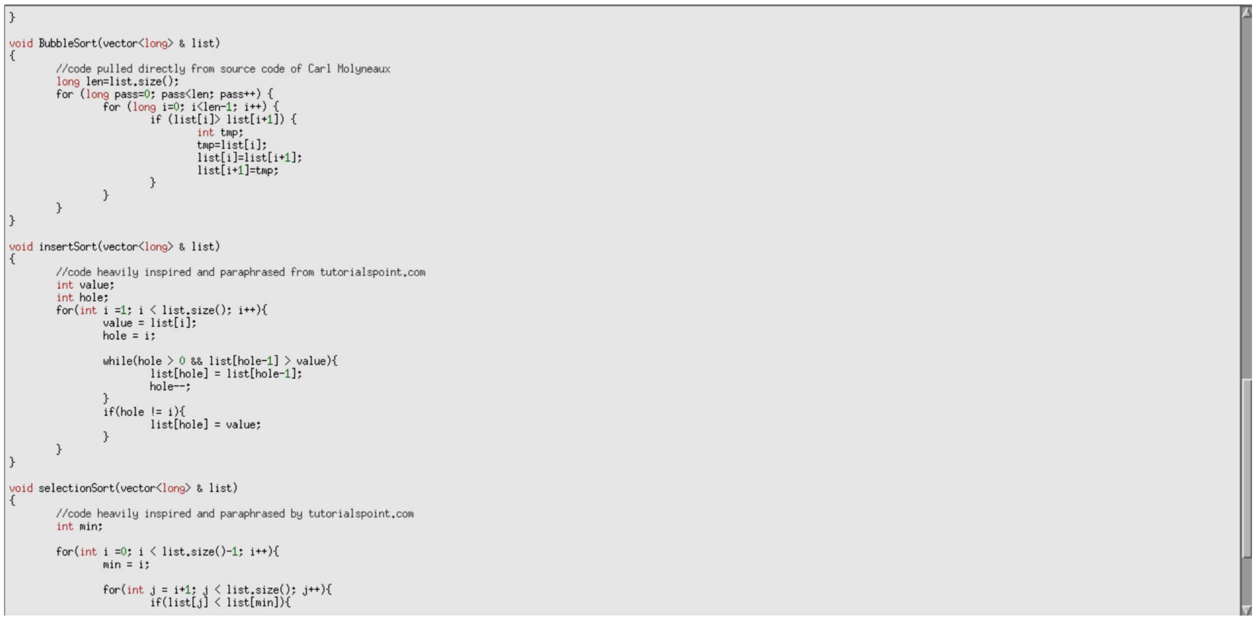
Source Code:


sortTimes.C - /home/students/e_prevost/homeworkAssignments/
File Edit Search Preferences Shell Macro Windows
Help

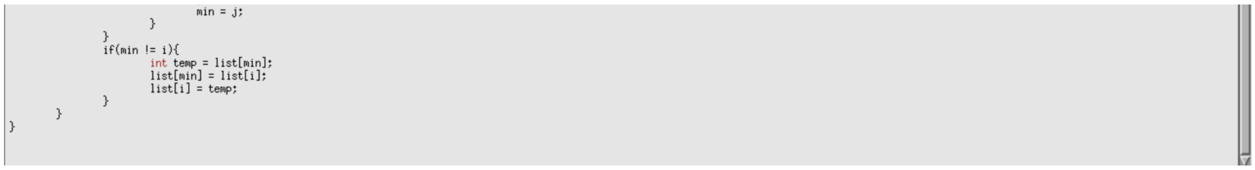
```
#include <iostream>
#include <climits>
#include <string>
#include <vector>
#include <ctime>
#include <cstdlib>

using namespace std;
//template portion pulled from templateInput.h from Carl Molyneaux
template<typename inputType>
inputType ReadValue(std::string prompt)
{
        inputType returnValue=0;
        std::cout << prompt;
        std::cin >> returnValue;
        while (std::cin.fail()==1) {
                std::cout << "Error! Cannot read input.\n";
                std::cin.clear();
                std::cin.ignore(INT_MAX,'\n');
                std::cout << prompt;
                std::cin >> returnValue;
        }
        return returnValue;
}


template<typename inputType>
inputType ReadValue(std::string prompt, inputType minValue)
{
        inputType returnValue=0;
        returnValue=ReadValue<inputType>(prompt);
        while (returnValue < minValue) {
                std::cout << "Error!  Value must be >= " << minValue << std::endl;
                returnValue=ReadValue<inputType>(prompt);
        }
        return returnValue;
}


template<typename inputType>
inputType ReadValue(std::string prompt, inputType minValue, inputType maxValue)
{
        inputType returnValue=0;
        returnValue=ReadValue<inputType>(prompt,minValue);
        while (returnValue > maxValue) {
                std::cout << "Error!  Value must be <= " << maxValue << std::endl;
                returnValue=ReadValue<inputType>(prompt,minValue);
```



```
        }
        return returnValue;
}
//template portion ends here
void Input(vector<long>& list);
void BubbleSort(vector<long> & list);
void insertSort(vector<long> & list);
void selectionSort(vector<long> & list);
int main()
{
        srand(time(nullptr));
        vector<long> x1;
        Input(x1);

        vector<long>x2,x3;
        x2 = x1;
        x3 = x1;

        long startTime = time(nullptr);
        BubbleSort(x1);
        long endTime = time(nullptr);
        long BStime = endTime-startTime;

         startTime = time(nullptr);
        insertSort(x2);
         endTime = time(nullptr);
        long IStime = endTime-startTime;

         startTime = time(nullptr);
        selectionSort(x3);
         endTime = time(nullptr);
        long SStime = endTime-startTime;
        cout<<"Bubble Sort Sorting time: ";
        cout<< BStime << endl;
        cout<<"Insert Sort Sorting time: ";
        cout<< IStime << endl;
        cout<<"Selection Sort Sorting time: ";
        cout<< SStime << endl;
}

void Input(vector<long>& list)
{
        //code pulled directly from source code of Carl Molyneaux
        long count=ReadValue<long>("How many values? ",0);
        for (long i=0; i<count; i++) {
                list.push_back(rand() % 10001);
        }
```



```
}

void BubbleSort(vector<long> & list)
{
        //code pulled directly from source code of Carl Molyneaux
        long len=list.size();
        for (long pass=0; pass<len; pass++) {
                for (long i=0; i<len-1; i++) {
                        if (list[i]> list[i+1]) {
                                int tmp;
                                tmp=list[i];
                                list[i]=list[i+1];
                                list[i+1]=tmp;
                        }
                }
        }
}

void insertSort(vector<long> & list)
{
        //code heavily inspired and paraphrased from tutorialspoint.com
        int value;
        int hole;
        for(int i =1; i < list.size(); i++){
                value = list[i];
                hole = i;

                while(hole > 0 && list[hole-1] > value){
                        list[hole] = list[hole-1];
                        hole--;
                }
                if(hole != i){
                        list[hole] = value;
                }
        }
}

void selectionSort(vector<long> & list)
{
        //code heavily inspired and paraphrased by tutorialspoint.com
        int min;

        for(int i =0; i < list.size()-1; i++){
                min = i;

                for(int j = i+1; j < list.size(); j++){
                        if(list[j] < list[min]){
```



```
                                min = j;
                        }
                }
                if(min != i){
                        int temp = list[min];
                        list[min] = list[i];
                        list[i] = temp;
                }
        }
}
```