Name: Amuldeep Dhillon
Class: CS 116-02 Sha 2017 Spring
Assignment: Lab 3-Wine Lists
Date: 3/19/2017

Description:
The Program will read a file of different Wines with descriptions. It will display the types of wine that fall within a user inputted price range along with the average price. Then do the same for a rating range. It will then display all the wine ranked by price lowest to highest, then once again the same thing for ratings. Lastly, it will display all the Red wines along with the lowest, highest, and average price. Then the same for White wines in the file.

Inputs:
- ❖ Wine Document
- ❖ Lowest Price
- ❖ Highest Price
- ❖ Lowest Rating
- ❖ Highest Rating

Outputs:
- ❖ My Name, Class, and current Time
- ❖ Ask user for price range
- ❖ The wines between the price range along with its type, vintage, rating, price, and city arranged in a table with labels along with their average price and the number of options
- ❖ Ask user for rating range
- ❖ The wines between the rating range along with its type, vintage, rating, price, and city arranged in a table with labels along with their average price and the number of options
- ❖ All the wines, with details, ranked lowest price to highest in a table with labels
- ❖ All the wines, with details, ranked lowest score to highest in a table with labels
- ❖ Just the red wines, with details in a table with labels, arranged by price. Along with the lowest, highest, and average price
- ❖ Just the white wines, with details in a table with labels, arranged by year. Along with the lowest, highest, and average price

Source Code:

Makefile:

```
# begin of Makefile
```

```
#
CC=g++ -std=c++11
#
CFLAGS = -c -Wall -I/usr/include/mysql
#LFLAGS = -L/usr/lib/mysql -lmysqlclient
LFLAGS =

all: lab

lab: printMeFirst.o readWine.o address.o getCity.o getPrice.o \
getRating.o getWineName.o getWineType.o getYear.o setCity.o setInfo.o \
setWineInfo.o setWineryName.o trimWords.o wine.o setAddress.o \
printInfo.o printMoreInfo.o scoreSort.o priceSort.o redWineSort.o \
whiteWineSort.o printLabels.o
   $(CC) printMeFirst.o readWine.o address.o getCity.o getPrice.o \
   getRating.o getWineName.o getWineType.o getYear.o setCity.o \
   setInfo.o setWineInfo.o setWineryName.o trimWords.o wine.o \
   setAddress.o printInfo.o printMoreInfo.o scoreSort.o priceSort.o \
   redWineSort.o whiteWineSort.o printLabels.o -o lab $(LFLAGS)

printMeFirst.o: printMeFirst.cpp
   $(CC) $(CFLAGS) printMeFirst.cpp

readWine.o: readWine.cpp
   $(CC) $(CFLAGS) readWine.cpp

address.o: address.cpp
   $(CC) $(CFLAGS) address.cpp

getCity.o: getCity.cpp
   $(CC) $(CFLAGS) getCity.cpp

getPrice.o: getPrice.cpp
   $(CC) $(CFLAGS) getPrice.cpp

getRating.o: getRating.cpp
   $(CC) $(CFLAGS) getRating.cpp

getWineName.o: getWineName.cpp
```

```
   $(CC) $(CFLAGS) getWineName.cpp

getWineType.o: getWineType.cpp
   $(CC) $(CFLAGS) getWineType.cpp

getYear.o: getYear.cpp
   $(CC) $(CFLAGS) getYear.cpp

setCity.o: setCity.cpp
   $(CC) $(CFLAGS) setCity.cpp

setInfo.o: setInfo.cpp
   $(CC) $(CFLAGS) setInfo.cpp

setWineInfo.o: setWineInfo.cpp
   $(CC) $(CFLAGS) setWineInfo.cpp

setWineryName.o: setWineryName.cpp
   $(CC) $(CFLAGS) setWineryName.cpp

trimWords.o: trimWords.cpp
   $(CC) $(CFLAGS) trimWords.cpp

wine.o: wine.cpp
   $(CC) $(CFLAGS) wine.cpp

setAddress.o: setAddress.cpp
   $(CC) $(CFLAGS) setAddress.cpp

printInfo.o: printInfo.cpp
   $(CC) $(CFLAGS) printInfo.cpp

printMoreInfo.o: printMoreInfo.cpp
   $(CC) $(CFLAGS) printMoreInfo.cpp

scoreSort.o: scoreSort.cpp
   $(CC) $(CFLAGS) scoreSort.cpp

priceSort.o: priceSort.cpp
```

```
   $(CC) $(CFLAGS) priceSort.cpp

redWineSort.o: redWineSort.cpp
   $(CC) $(CFLAGS) redWineSort.cpp

whiteWineSort.o: whiteWineSort.cpp
   $(CC) $(CFLAGS) whiteWineSort.cpp

printLabels.o: printLabels.cpp
   $(CC) $(CFLAGS) printLabels.cpp
clean:
   rm *.o lab

run:
   ./lab

#end of makefile
```

__Winelist.txt:__ the actual document with all the data

```
Stags Leap Artemis Cabernet;Red;2013;92;65;Stags; 6150 Silverado Trail; Napa; CA; 94558
Silver Oak Cabernet;Red;2011;91;110;Silver Oak; 915 Oakville Cross Rd; Oakville; CA ; 94562
Joseph Phelps Insignia;Red;2013;97;240;Joseph Phphelps; 200 Taplin Road ; St. Helena; CA;
94574
Duckhorn Cabernet;Red;2013;93;72;Duckhorn ; 1000 Lodi Lan; St. Helena; CA; 94574
Alpha Omega Chardonnay;White;2012;92;69.99;Alpha Omega ; 1155 Mee Lane at Hwy 29; St.
Helena; CA; 94574
Grgich Chardonnay;White;2013;90;43;Grgich ; 1829 St. Helena Hwy; Rutherford; CA; 94573
Stags Leap Chardonnay;White;2014;90;30;Stags; 6150 Silverado Trail; Napa; CA; 94558
Pahlmeyer;White;2013;93;72.99;Pahlmeyer ; 811 St. Helena Hwy; St. Helena; CA; 94574
```

__readWine.cpp:__

```
#include "wine.h"


using namespace std;

const char SPLIT_CHAR = ';';
```

```
/*
 * Purpose: Sort the Document info into a newly
 * made vector, sort and print wines in order of both price and rating
 *  and execute other functions
 *
 * @author: Amuldeep Dhillon
 * @version: 1.0 3/7/2016
 *
 * @param:Wine Document
 *
 * @Outputs: Labels to tell user when each program is beginning
 *
 * @return:0
*/

int main(int argc, char* argv[])
{
   printMeFirst("Amuldeep Dhillon", "Lab 3: CS 116-02 Thursdays");
         string str1, str;
         vector < string > tokens;
   string wineName, type, wineryName, address, city, state;
   int vintage, rating, zipcode;
   double price;
   string fileName;

   vector<Wine*> fptr;

   Wine *f1;

   Address addressDetails;

  // use filename if provided in the parameter list
         if (argc < 2)
   {
          cout <<"Usage: " << argv[0] << " input_file\n";
          cout <<"Using default file winelist.txt instead\n";
          fileName = "winelist.txt";
   }
   else
          fileName = argv[1];


  ifstream myfile (fileName.c_str()); // open the file


  if (myfile.is_open()) {
         while (myfile) {
         if (!getline(myfile, str))
         break; //end of file

         istringstream split(str);
         //  for (string each; getline(split, each, split_char); tokens.push_back(each));
         // OR USE THE WHILE LOOP BELOW
         vector <string> tokens;
         while (split) // parse the line
         {
         string s;
         if (!getline(split, s, SPLIT_CHAR))
         break; // end of line
         else
         {
         str1 = trimWords(s);
         tokens.push_back(str1);
         }
         }
         // now use `tokens`

         for (unsigned int i = 0; i < tokens.size(); i++)
         {
          switch (i)
          {
                 case 0:
                         wineName = tokens[0];
                         break;
                 case 1:
                         type= tokens[1];
```

```
                    break;
            case 2:
                    vintage = stoi(tokens[2]);
                    break;
            case 3:
                    rating = stoi(tokens[3]);
                    break;
            case 4:
                    price= stod(tokens[4]);
                    break;
            case 5:
                    wineryName = tokens[5];
                    break;
            case 6:
                    address = tokens[6];
                    break;
            case 7:
                    city = tokens[7];
                    break;
            case 8:
                    state = tokens[8];
                    break;
            case 9:
                    zipcode = stoi(tokens[i]);
                    break;
        }
       }

        f1 = new Wine(wineName,type,vintage,rating,price);
        addressDetails.setCity(city);
        f1->setAddress(addressDetails);
        fptr.push_back(f1);

       }
scoreSort(fptr);
priceSort(fptr);
std::cout << "\nRanking Based On Price\n" << std::endl;
printLabels();
sort(fptr.begin(), fptr.end(), ComparePrice());
```

```
    for (unsigned int i=0; i < fptr.size(); i++)
    {
            fptr[i]->printInfo();
    }
    std::cout << "\nRanking Based On Rating\n" << std::endl;
    printLabels();
    sort(fptr.begin(), fptr.end(), CompareScore());
    for (unsigned int i=0; i < fptr.size(); i++)
    {
            fptr[i]->printInfo();
    }
    redWineSort(fptr);
    whiteWineSort(fptr);
           myfile.close();
  }
  else
           cout << "Unable to open file";



  return 0;
}
```

Wine.h:

```
#ifndef WINE_H
#define WINE_H

/*
 *
 * Function documentations are located at the function definition
 * not their headings
 *
*/

#include <fstream>
#include <sstream>
#include <iostream>
#include <string>
```

```
#include <vector>
#include <algorithm>
#include <iomanip>
#include <ctime>

void printMeFirst(std::string name, std::string courseInfo);

std::string trimWords(const std::string & sentence);

using namespace std;

/*
 *
 *Class Structure for Address
 *
*/

class Address

{
public:

   Address();

   Address(string winery_name, string winery_street,

            string winery_city, string winery_state, int winery_zip) ;

   void setInfo(string winery_name, string winery_street,

            string winery_city, string winery_state,  int winery_zip) ;

   void setWineryName (string winery_name);

   void setCity (string c);

   string getCity();

private:
    string wineryName, street, city, state;

    int zipCode;

};

/*
 *
 *Class Structure for Wine
 *
*/

class Wine

{
public:

/*
 * Purpose: Sets paddress pointer to a new Address
 *
 * @author: Amuldeep Dhillon
 * @version: 1.0 3/7/2016
 *
 * @param:none
 *
 * @return:none
*/
    Wine() {  paddress=new Address();  }

    Wine(string wine_name,string wine_type, int wine_year,

            int wine_rating, double wine_price);

    void setInfo(string wine_name,string wine_type, int wine_year,

            int wine_rating, double wine_price);

/*
```

```
 * Purpose: Delete the new Address we originally created
 *
 * @author: Amuldeep Dhillon
 * @version: 1.0 3/7/2016
 *
 * @param:none
 *
 * @return:none
*/

  ~Wine() { if (paddress != NULL)

          {delete paddress; paddress = NULL;}  }

          void setAddress(Address a) ;

          int getRating();

          double getPrice();

          string getWineName();

          string getWineType();

          int getYear();

          void printInfo();

/*
 * Purpose:
 *
 * @author: Amuldeep Dhillon
 * @version: 1.0 3/7/2016
 *
 * @param:Wine type by reference
 *
 * @return:
 * true if first wine year is lower than the second's
 * false if first wine year is higher than the second's
```

```
*/

            bool operator< (const Wine& p2) const
              { return this->wineYear < p2.wineYear; }


    private:

            string wineName, wineType;

            int wineYear, wineRating;

            double winePrice;

            Address *paddress;

};

/*
 * Class structure of CompareScore
*/

class CompareScore{
    public:

/*
 * Purpose: Compares
 *
 * @author: Amuldeep Dhillon
 * @version: 1.0 3/7/2016
 *
 * @param:Wine pointer
 * @param:Another Wine pointer
 *
 * @return:
 * true if the first wine pointer's rating is lower than the second's
 * false if the first wine pointer's rating is higher than the second's
 *
*/
```

```
            bool operator() (Wine *a, Wine *b)
            {
                    return a->getRating() < b->getRating();
            }
};

/*
 *Class structure for Compare price
*/

class ComparePrice{
   public:
/*
 * Purpose:
 *
 * @author: Amuldeep Dhillon
 * @version: 1.0 3/7/2016
 *
 * @param:Wine pointer
 * @param:Another wine pointer
 *
 * @return:
 * true if first wine pointer's price is lower than the second's
 * false if first wine pointer's price is higher than the second's
*/

            bool operator() (Wine *a, Wine *b)
            {
                    return a->getPrice() < b->getPrice();
            }
};



void scoreSort(std::vector<Wine*>& test);
void priceSort(std::vector<Wine*>& test);
void redWineSort(std::vector<Wine*>& test);
void whiteWineSort(std::vector<Wine*>& test);
void printLabels();

#endif
```

printMeFirst.cpp:

```
/*
 *Purpose:
 * Print out programmer's information such as name, class information
 * and date/time the program is run
 *
 * @author Ron Sha
 * @version 1.0 1/27/2017
 *
 * @param name - Amuldeep Dhillon
 * @param courseInfo - Lab 3: CS 116-02 Thursdays
 * @return - none
*/


#include "wine.h"
void printMeFirst(std::string name, std::string courseInfo)

{

std::cout <<" Program written by: "<< name << std::endl; // put your name here

std::cout <<" Course info: "<< courseInfo << std::endl;

time_t now = time(0);  // current date/time based on current system

char* dt = ctime(&now); // convert now to string for

  std::cout << " Date: " << dt << std::endl;

}
```

trimWords.cpp:

```
/*
 * Purpose: Remove the empyt white spaces between data elements
 *
 * @author: Amuldeep Dhillon
 * @version: 1.0 3/7/2016
 *
 * @param: string of data from the document
 *
 *
 * @return:the data element put in without space "string"
*/


#include "wine.h"

using namespace std;

string trimWords(const string & sentence) {
  stringstream ss;
  string s;
  string out;

  ss << sentence;
  while (ss >> s)
  {
          out += (s + ' ');
  }
  return out.substr(0, out.length() - 1);
}
```

Address.cpp:

```
#include "wine.h"

/*
 * Purpose:set addresses city to blank
 *
 * @author: Amuldeep Dhillon
 * @version: 1.0 3/7/2016
 *
 * @param:none
 *
 * @return:none
*/


Address::Address(){
   city = "";
}

/*
 * Purpose:set WineryName, street, city, state, zipcode respectively
 * to the parameters from the document
 *
 * @author: Amuldeep Dhillon
 * @version: 1.0 3/7/2016
 *
 * @param:name of the winery
 * @param:name of the winery street
 * @param:name of the winery city
 * @param:name of the winery state
 * @param:the winery's zipcode
 *
 *
 * @return:none
*/


Address::Address(string winery_name, string winery_street,
                        string winery_city, string winery_state, int winery_zip){

   wineryName = winery_name;
   street = winery_street;
   city = winery_city;
   state = winery_state;
```

```
    zipCode = winery_zip;
}
```

setInfo.cpp:

```
#include "wine.h"

/*
 * Purpose:set wineryName, street, city, state, and zipcode manually
 * by each respective parameter
 *
 * @author: Amuldeep Dhillon
 * @version: 1.0 3/7/2016
 *
 * @param:name of the Winery
 * @param:name of the Winery street
 * @param:name of the Winery city
 * @param:name of the Winery state
 * @param:Winery zipcode
 *
 *
 * @return:none
*/


void Address::setInfo(string winery_name, string winery_street,
                      string winery_city, string winery_state,  int winery_zip){

    wineryName = winery_name;
    street = winery_street;
    city = winery_city;
    state = winery_state;
    zipCode = winery_zip;
}
```

setWineryName.cpp:

```
#include "wine.h"
```

```
/*
 * Purpose:sets the Winery Name
 *
 * @author: Amuldeep Dhillon
 * @version: 1.0 3/7/2016
 *
 * @param:Name of the winery
 *
 * @return:none
*/


void Address::setWineryName (string w){
    wineryName = w;
}
```

setCity.cpp:

```
#include "wine.h"

/*
 * Purpose:Set Winery's City
 *
 * @author: Amuldeep Dhillon
 * @version: 1.0 3/7/2016
 *
 * @param:Name of the Winery's City
 *
 * @return:none
*/


void Address::setCity (string c){
    city = c;
}
```

getCity.cpp:

```
#include "wine.h"
```

```
/*
 * Purpose:return the Winery's City
 *
 * @author: Amuldeep Dhillon
 * @version: 1.0 3/7/2016
 *
 * @param:none
 *
 * @return:the Winery's City
*/


string Address::getCity(){
    return city;
}
```

__Wine.cpp:__

```
#include "wine.h"

/*
 * Purpose:sets the wine name, type, year, rating, price, and address
 * pointer respective to the parameters from the document
 *
 * @author: Amuldeep Dhillon
 * @version: 1.0 3/7/2016
 *
 * @param:name of the wine
 * @param:the wine type
 * @param:the vintage of the wine
 * @param:the wine's rating
 * @param:the wine's price
 *
 * @return:none
*/


Wine::Wine(string wine_name,string wine_type, int wine_year,
        int wine_rating, double wine_price){
                wineName = wine_name;
                wineType = wine_type;
                wineYear = wine_year;
                wineRating = wine_rating;
                winePrice = wine_price;
                paddress= new Address;
}
```

setWineInfo.cpp:

```
#include "wine.h"

/*
 * Purpose:sets the wine's name, type, year, rating, and price manually
 * respective to each parameter
 *
 * @author: Amuldeep Dhillon
 * @version: 1.0 3/7/2016
 *
 * @param:wine's name
 * @param:wine's type
 * @param:wine's vintage
 * @param:wine's rating
 * @param:wine's price
 *
 * @return:none
*/


void Wine::setInfo(string wine_name,string wine_type, int wine_year,
        int wine_rating, double wine_price){
                wineName = wine_name;
                wineType = wine_type;
                wineYear = wine_year;
                wineRating = wine_rating;
                winePrice = wine_price;
}
```

setAddress.cpp:

```
#include "wine.h"

/*
 * Purpose:sets the wine's address ponter
 *
 * @author: Amuldeep Dhillon
 * @version: 1.0 3/7/2016
 *
 * @param:The address of the wine
 *
 *
 * @return:none
*/


void Wine::setAddress(Address a){
    *paddress = a;
}
```

getRating.cpp:

```
#include "wine.h"
/*
 * Purpose:get the wine's rating
 *
 * @author: Amuldeep Dhillon
 * @version: 1.0 3/7/2016
 *
 * @param:none
 *
 * @return:wine's rating
*/

int Wine::getRating () {
    return wineRating;
}
```

getPrice.cpp:

```
#include "wine.h"

/*
 * Purpose:retrieve wine's price
 *
 * @author: Amuldeep Dhillon
 * @version: 1.0 3/7/2017
 *
 * @param:none
 *
 * @return:
*/


double Wine::getPrice(){
   return winePrice;
}
```

getWineName.cpp:

```
#include "wine.h"

/*
 * Purpose:retrive wine's name
 *
 * @author: Amuldeep Dhillon
 * @version: 1.0 3/7/2016
 *
 * @param:none
 *
 * @return:the wine's name
*/


string Wine::getWineName(){
   return wineName;
}
```

getWineType.cpp:

```
#include "wine.h"

/*
 * Purpose:retrieve Wine's type
 *
 * @author: Amuldeep Dhillon
 * @version: 1.0 3/7/2016
 *
 * @param:none
 *
 * @return:the wine's type
*/


string Wine::getWineType(){
   return wineType;
}
```

getYear.cpp:

```
#include "wine.h"

/*
 * Purpose:retrieve the wine's vintage
 *
 * @author: Amuldeep Dhillon
 * @version: 1.0 3/7/2016
 *
 * @param:none
 *
 * @return:the wine's vintage
*/


int Wine::getYear(){
    return wineYear;
}
```

printInfo.cpp:

```
#include "wine.h"

/*
 * Purpose:displays all the info on a wine
 *
 * @author: Amuldeep Dhillon
 * @version: 1.0 3/7/2016
 *
 * @param:none
 *
 * @Outputs:The wine's name, type, year, rating, price, and city of
 *                    origin with organized spaces
 *
 * @return:none
*/


void Wine::printInfo(){
    string city;
    if (paddress != NULL)
            city=paddress->getCity();
    else
            city = "None";

    cout << left << setw(30) << setfill(' ') << wineName << setw(15) << setfill(' ')
            << wineType << setw(15) << setfill(' ') << wineYear << setw(15) << setfill(' ')
            << wineRating << setw(15) << setfill(' ') << winePrice << setw(15) << setfill(' ') << city
<<endl;
}
```

scoreSort.cpp:

```
#include "wine.h"
```

```
/*
 * Purpose:display all the wine's within an enter rating range
 *
 * @author: Amuldeep Dhillon
 * @version: 1.0 3/7/2016
 *
 * @param:the vector fptr, that holds all the wine pointers, from main
 *
 * @Inputs: Bottom rating, Top rating
 * @Outputs:
 *                  Notice that score sorting has begun
 *                  ask user for score range lowest then highest
 *                  Error message if they do not insert a positive number
 *                  Notice that they have inserted the numbers back ward and
 *                          flip them
 *                  All the wine in the range
 *                  Number of options
 *                  Average price of the wine
 *
 * @return:none
*/


void scoreSort(std::vector<Wine*>& test){
   int input1;
   int input2;
   int count = 0;
   double priceSum = 0;
   double priceAverage = 0;
   std::cout << "Beginning the Score Sorting \n";
   sort(test.begin(), test.end(), CompareScore());
   std::cout << "Enter a Bottom Score: ";
   std::cin >> input1;
   while(cin.fail() || input1 < 0){
         cin.clear();
         cin.ignore(numeric_limits<streamsize>::max(), '\n');
         cout << "Error: Please Enter A Positive Number" << endl;
         cin >> input1;
   }
```

```
    std::cout << "Enter a Top Score: ";
    std::cin >> input2;
    while(cin.fail() || input2 < 0){
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            cout << "Error: Please Enter A Positive Number" << endl;
            cin >> input2;
    }
    if(input1 > input2){
            std::cout << "You have entered the numbers backwards, We are "
            << "switching the inputs";
            int temp = input2;
            input2 = input1;
            input1 = temp;
    }
    printLabels();
    for (unsigned int i=0; i < test.size(); i++)
    {
            if(test[i]->getRating() >= input1 && test[i]->getRating() <= input2)
                    {test[i]->printInfo();
                    priceSum = priceSum + test[i]->getPrice();
                    count++;}
    }
    priceAverage = (priceSum/count);
    string c = std::to_string(count);
    string pA = std::to_string(priceAverage);
    std::cout << fixed << setprecision(2) << "\nThere are " << c << " options" << std::endl;
    std::cout <<  "The Average price is " << pA << std::endl;
}
```

priceSort.cpp:

```
#include "wine.h"

/*
 * Purpose:sort Wine's that fall within the user entered price range
 *
 * @author: Amuldeep Dhillon
 * @version: 1.0 3/7/2016
 *
 * @param: the vector fptr that holds all the wine pointers
 *
 * @Inputs:Lowest price, Highest price
 * @Outputs:Notice that the price sorting is beginning
 *                  Ask user for price range
 *                  Error message if they do not insert a positive number
 *                  Notice the user if they inserted the range backwards
 *                          and are switching the inputs
 *                  All the wines that fall in the price range with their info
 *                  Number of options
 *                  Average price of all the ranged wine
 *
 * @return:none
*/


void priceSort(std::vector<Wine*>& test){
   int input1;
   int input2;
   int count = 0;
   double priceSum = 0;
   double priceAverage = 0;
   std::cout << "\n\nBeginning the Price Sorting \n";
   sort(test.begin(), test.end(), ComparePrice());
   std::cout << "Enter a Bottom Price: ";
   std::cin >> input1;
   while(cin.fail() || input1 < 0){
         cin.clear();
         cin.ignore(numeric_limits<streamsize>::max(), '\n');
         cout << "Error: Please Enter A Positive Number" << endl;
         cin >> input1;
   }
   std::cout << "Enter a Top Price: ";
   std::cin >> input2;
   while(cin.fail() || input2 < 0){
         cin.clear();
```

```
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            cout << "Error: Please Enter A Positive Number" << endl;
            cin >> input2;
    }
    if(input1 > input2){
            std::cout << "You have entered the numbers backwards, We are "
            << "switching the inputs";
            int temp = input2;
            input2 = input1;
            input1 = temp;
    }
    printLabels();
    for (unsigned int i=0; i < test.size(); i++)
    {
            if(test[i]->getPrice() >= input1 && test[i]->getPrice() <= input2)
                    {test[i]->printInfo();
                    priceSum = priceSum + test[i]->getPrice();
                    count++;}
    }
    priceAverage = (priceSum/count);
    string c = std::to_string(count);
    string pA = std::to_string(priceAverage);
    std::cout << "\nThere are " << c << " options" << std::endl;
    std::cout <<  "The Average price is " << pA << std::endl;
}
```

redWineSort.cpp:

```
#include "wine.h"

/*
 * Purpose:display all the Red Wines sorted by Price
 *
 * @author: Amuldeep Dhillon
 * @version: 1.0 3/7/2016
 *
 * @param:the vector fptr that holds all the wine pointers
 *
 *
 * @Outputs:Notice that they are display just the red wines
 *                  display all the red wine's and their information
 *                  the lowest price
 *                  the highest price
 *                  the average price
 *
 * @return:none
*/


void redWineSort(std::vector<Wine*>& test){
   vector<double> Redwine;
   int count = 0;
   double priceSum = 0;
   double priceAverage = 0;
   sort(test.begin(), test.end(), ComparePrice());
   std::cout << "\n\nJust Red Wine\n";
   printLabels();
   for (unsigned int i=0; i < test.size(); i++)
   {if(test[i]->getWineType() == "Red"){
           test[i]->printInfo();
           priceSum = priceSum + test[i]->getPrice();
           Redwine.push_back(test[i]->getPrice());
           count++;
   }
   }
   string lP = std::to_string(Redwine[0]);
   string hP = std::to_string(Redwine[Redwine.size()-1]);
   std::cout << "\nThe Lowest Price is: " << lP << std::endl;
   std::cout << "The Highest Price is: " << hP << std::endl;
   priceAverage = priceSum/count;
   string pA = std::to_string(priceAverage);
   std::cout << "The Average Price is: " << pA <<std::endl;
}
```

<u>whiteWineSort.cpp:</u>

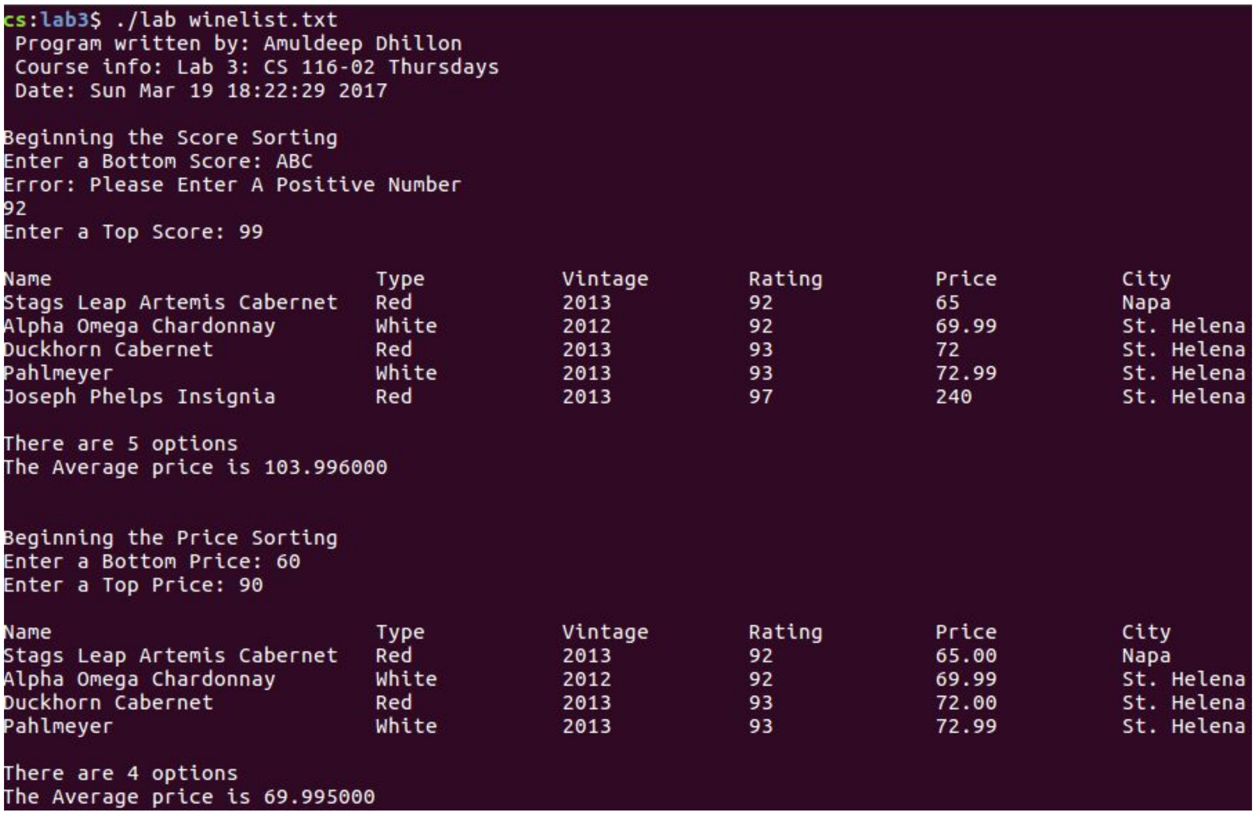
```
#include "wine.h"

/*
 * Purpose:display all the White Wines sorted by Price
 *
 * @author: Amuldeep Dhillon
 * @version: 1.0 3/7/2016
 *
 * @param:the vector fptr that holds all the wine pointers
 *
 *
 * @Outputs:Notice that they are display just the white wines
 *                  display all the white wine's and their information
 *                  the lowest price
 *                  the highest price
 *                  the average price
 *
 * @return:none
*/


void whiteWineSort(std::vector<Wine*>& test){
   vector<double> Whitewine;
   int count = 0;
   double priceSum = 0;
   double priceAverage = 0;
   sort(test.begin(), test.end());
   std::cout << "\n\nJust White Wine\n";
   printLabels();
   for (unsigned int i=0; i < test.size(); i++)
   {if(test[i]->getWineType() == "White"){
          test[i]->printInfo();
          priceSum = priceSum + test[i]->getPrice();
          Whitewine.push_back(test[i]->getPrice());
          count++;
   }
   }
```

```
    sort(Whitewine.begin(), Whitewine.end());
    string lP = std::to_string(Whitewine[0]);
    string hP = std::to_string(Whitewine[Whitewine.size()-1]);
    std::cout << "\nThe Lowest Price is: " << lP << std::endl;
    std::cout << "The Highest Price is: " << hP << std::endl;
    priceAverage = priceSum/count;
    string pA = std::to_string(priceAverage);
    std::cout << "The Average Price is: " << pA <<std::endl;
}
```

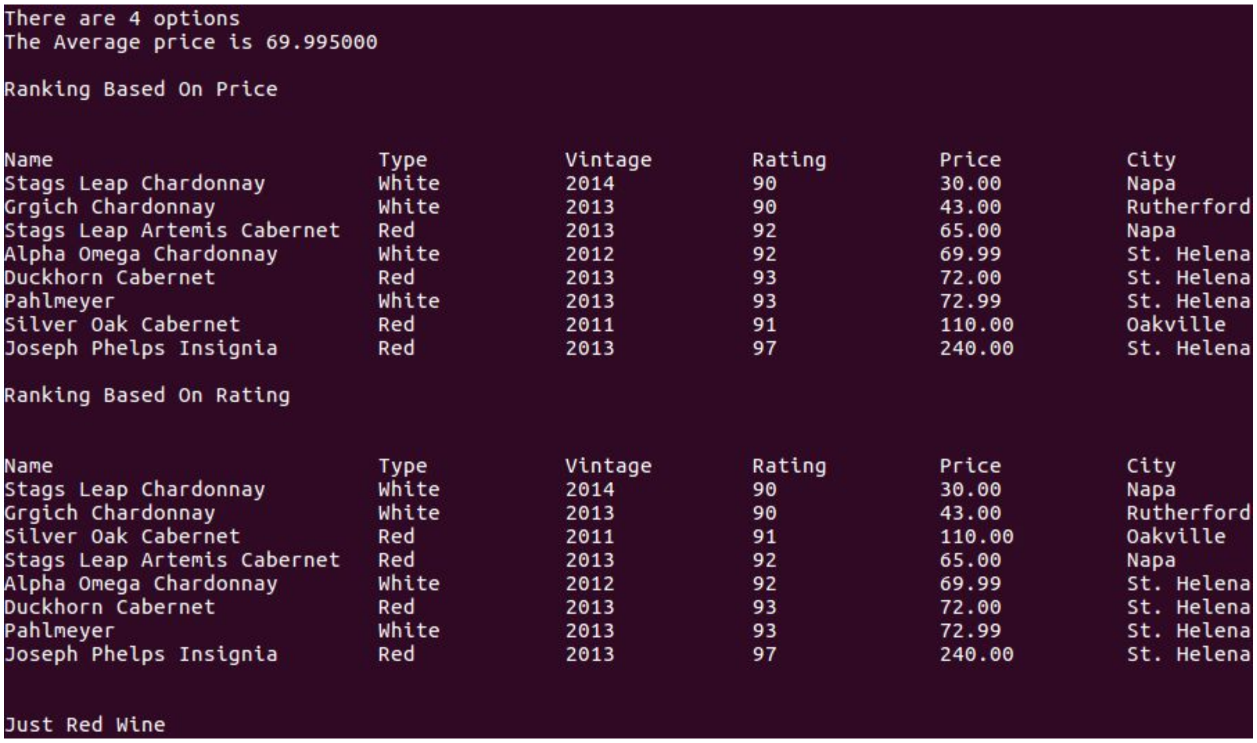
printLabels.cpp:

```
#include "wine.h"

/*
 * Purpose:print the labels for all the wine lists
 *
 * @author: Amuldeep Dhillon
 * @version: 1.0 3/7/2016
 *
 * @param:none
 *
 *
 * @Outputs:The words name, type, vintage, rating, price, and city in
 *                   the same location as the list will appear in
 *
 * @return:none
*/



void printLabels(){
    std::cout << "\n";
    std::cout << left << setw(30) << setfill(' ') << "Name" << setw(15)
    << setfill(' ') << "Type" << setw(15) << setfill(' ') << "Vintage"
    << setw(15) << setfill(' ') << "Rating" << setw(15) << setfill(' ')
    << "Price" << setw(15) << setfill(' ') << "City" <<std::endl;
}
```

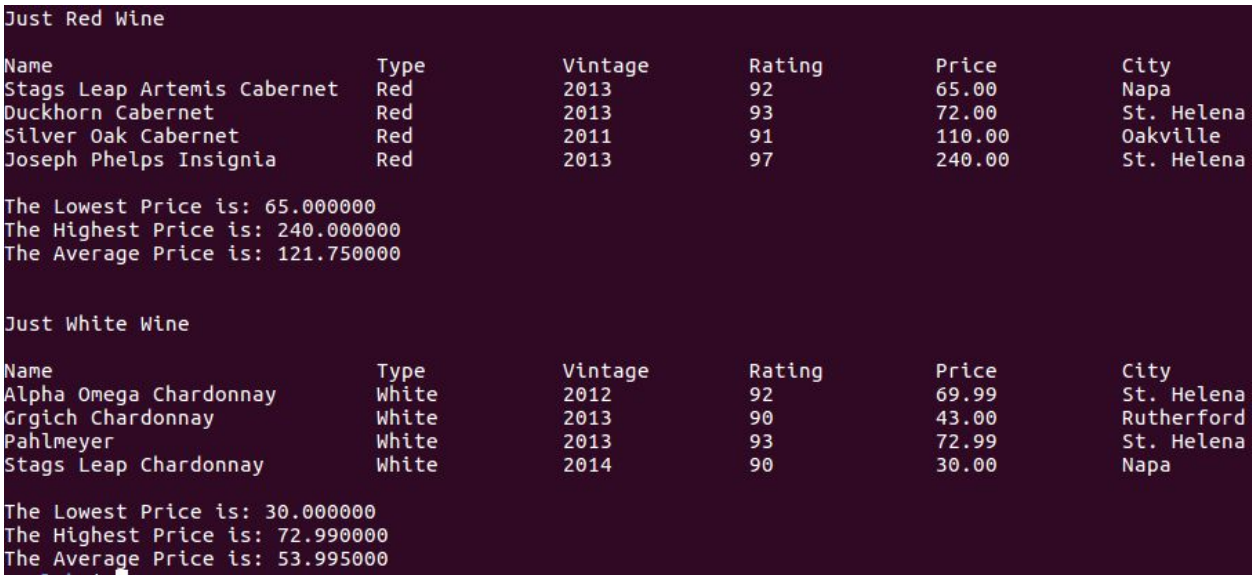
Screen Shots:



```
cs:lab3$ ./lab winelist.txt
 Program written by: Amuldeep Dhillon
 Course info: Lab 3: CS 116-02 Thursdays
 Date: Sun Mar 19 18:22:29 2017

Beginning the Score Sorting
Enter a Bottom Score: ABC
Error: Please Enter A Positive Number
92
Enter a Top Score: 99

Name                          Type           Vintage        Rating         Price          City
Stags Leap Artemis Cabernet   Red            2013           92             65             Napa
Alpha Omega Chardonnay        White          2012           92             69.99          St. Helena
Duckhorn Cabernet             Red            2013           93             72             St. Helena
Pahlmeyer                     White          2013           93             72.99          St. Helena
Joseph Phelps Insignia        Red            2013           97             240            St. Helena

There are 5 options
The Average price is 103.996000


Beginning the Price Sorting
Enter a Bottom Price: 60
Enter a Top Price: 90

Name                          Type           Vintage        Rating         Price          City
Stags Leap Artemis Cabernet   Red            2013           92             65.00          Napa
Alpha Omega Chardonnay        White          2012           92             69.99          St. Helena
Duckhorn Cabernet             Red            2013           93             72.00          St. Helena
Pahlmeyer                     White          2013           93             72.99          St. Helena

There are 4 options
The Average price is 69.995000
```



```
There are 4 options
The Average price is 69.995000

Ranking Based On Price


Name                          Type           Vintage        Rating         Price          City
Stags Leap Chardonnay         White          2014           90             30.00          Napa
Grgich Chardonnay             White          2013           90             43.00          Rutherford
Stags Leap Artemis Cabernet   Red            2013           92             65.00          Napa
Alpha Omega Chardonnay        White          2012           92             69.99          St. Helena
Duckhorn Cabernet             Red            2013           93             72.00          St. Helena
Pahlmeyer                     White          2013           93             72.99          St. Helena
Silver Oak Cabernet           Red            2011           91             110.00         Oakville
Joseph Phelps Insignia        Red            2013           97             240.00         St. Helena

Ranking Based On Rating


Name                          Type           Vintage        Rating         Price          City
Stags Leap Chardonnay         White          2014           90             30.00          Napa
Grgich Chardonnay             White          2013           90             43.00          Rutherford
Silver Oak Cabernet           Red            2011           91             110.00         Oakville
Stags Leap Artemis Cabernet   Red            2013           92             65.00          Napa
Alpha Omega Chardonnay        White          2012           92             69.99          St. Helena
Duckhorn Cabernet             Red            2013           93             72.00          St. Helena
Pahlmeyer                     White          2013           93             72.99          St. Helena
Joseph Phelps Insignia        Red            2013           97             240.00         St. Helena


Just Red Wine
```



```
Just Red Wine

Name                          Type           Vintage        Rating         Price          City
Stags Leap Artemis Cabernet   Red            2013           92             65.00          Napa
Duckhorn Cabernet             Red            2013           93             72.00          St. Helena
Silver Oak Cabernet           Red            2011           91             110.00         Oakville
Joseph Phelps Insignia        Red            2013           97             240.00         St. Helena

The Lowest Price is: 65.000000
The Highest Price is: 240.000000
The Average Price is: 121.750000


Just White Wine

Name                          Type           Vintage        Rating         Price          City
Alpha Omega Chardonnay        White          2012           92             69.99          St. Helena
Grgich Chardonnay             White          2013           90             43.00          Rutherford
Pahlmeyer                     White          2013           93             72.99          St. Helena
Stags Leap Chardonnay         White          2014           90             30.00          Napa

The Lowest Price is: 30.000000
The Highest Price is: 72.990000
The Average Price is: 53.995000
```