```
Script started on 2022-09-27 15:07:48-05:00 [TERM="xterm" TTY="/dev/pts/9" COLUMNS=
j_pec2@ares:~/JPMainDir/CSC122/Port1/CityMapProject$ pwd
/home/students/j_pec2/JPMainDir/CSC122/Port1/CityMapProject
j_pec2@ares:~/JPMainDir/CSC122/Port1/CityMapProject$ cat MAPDriverJPInfo.txt.inf
o
cat: MAPDriverJPInfo.txt.info: No such file or directory
j_pec2@ares:~/JPMainDir/CSC122/Port1/CityMapProject$ cat MAPDriverJPInfo.txt.info


nfo
cat: MAPDriverJPInfo.info: No such file or directory
j_pec2@ares:~/JPMainDir/CSC122/Port1/CityMapProject$ cat MAPDriverJPInfo.txt
Jack Pec



CSC122-001



Map it out Project



Base level 4



Added (Level 3) to add a submenu to sort the city list.



Level 7



Desc:



It's the Map it out Project

with strings and vectors.

j_pec2@ares:~/JPMainDir/CSC122/Port1/CityMapProject$ show-cod


MAPDriverJP.cpp:
```

```
1  #include <iostream>
2  #include "Input_prot2.h"
3  #include "point.h"
4  #include "city.h"
5
6
7
8  //function prototypes
9
10 //////////
11 void menu(std::vector<City> &vec);
12 void distanceTwoCities(std::vector<City> &vec);
13 void cityInput(std::vector<City> &vec);
14 void citySort( std::vector<City> &vec);
15 void listoutput(std::vector<City> &vec);
16
17 bool insert(std::vector<City> & vec,
18             std::vector<City>::size_type before_me,
19             City new_item,
20             std::vector<City>::size_type end);
21
22 void insertion(std::vector<City> & vec,short typeOfSort);
23 ////////////
24
25
26 int main()
27 {
28
29     std::vector<City> cityList;
30     menu(cityList);
31
32     return 0;
33 }
34
35 ///////////////
36
37
38
39
40 void menu(std::vector<City> &vec){
41
42 bool valid = true;
43     while(valid){
44         std::cout << " 1) Enter city Information" << std::endl;
45         std::cout << " 2) calculate Distance between two cities"<< std::en
46         std::cout << " 3) Print All cities"<< std::endl;
47         std::cout << " 4) Sort the cities"<< std::endl;
48         std::cout << " 5) Quit"<< std::endl;
49
50         char menuChoice = input_protect<char>(" ");
51
52         if(menuChoice == '1'||menuChoice == 'E'||menuChoice == 'I'){
53             cityInput(vec);
54
55         }
56         else if(menuChoice == '2'||menuChoice == 'D'){
57             distanceTwoCities(vec);
58
```

```
        }
        else if(menuChoice == '3'
                ||menuChoice == 'P'
                ||menuChoice == 'A'){
            listoutput(vec);
        }
        else if(menuChoice == '4'||menuChoice == 'S'){
            citySort(vec);
        }
        else if(menuChoice == '5'||menuChoice == 'Q'){
            valid = false;

        }

    }



}

void distanceTwoCities(std::vector<City> &vec){
    size_t indexCity1;
    size_t indexCity2;
    double distance;

  if (vec.size() < 2) {
      std::cout
      << "Need 2 or more cities to find distance." <<std::endl;

  }
  else if(vec.size()< 3){
    distance = vec[0].distance(vec[1]);
    std::cout << "Distance between " << vec[0].get_name()
    << " and " << vec[1].get_name() << " is: "
    << distance << std::endl;

  }
  else if(vec.size() >=  3){
        listoutput(vec);
        std::cout << "Select 2 indexes for 2 Citys:" <<std::endl;
        indexCity1 = input_protect("Index 1: ",
                    1, static_cast<int>(vec.size()));
        indexCity2 = input_protect("Index 2: ",
                    1, static_cast<int>(vec.size()));



        while(indexCity1 == indexCity2){
        std::cout << "Cant pick the same city twice: ";
        indexCity1 = input_protect("Index 1: ",
                    1, static_cast<int>(vec.size()));
        indexCity2 = input_protect("Index 2: ",
                    1, static_cast<int>(vec.size()));



        }

        indexCity1 = indexCity1 -1;
        indexCity2 = indexCity2 -1;

        if(indexCity1 != indexCity2){
        distance = vec[indexCity1].distance(vec[indexCity2]);
        std::cout << "Distance between "
        << vec[indexCity1].get_name()
        << " and " << vec[indexCity2].get_name()
        << " is: " << distance <<std::endl;



        }


  }


}

void cityInput(std::vector<City> &vec){


    bool valid = true;
    while(valid){


        City cityToAdd;

        std::string input;
        std::cout << "City Name: ";
            if (std::cin.peek() == '\n')
            {
                std::cin.ignore();
            }
        std::getline(std::cin,input);



        cityToAdd.set_name(input);

       //  double x,y;

       //   x = input_protect<double>("X value: ");

      //    y = input_protect<double>("Y value: ");

        Point newP;
        newP.Input();
```

```
167
168         cityToAdd.set_location(newP);
169
170
171         vec.push_back(cityToAdd);
172
173       //  citySort(vec);
174
175         bool stillvalid = input_protectChoiceBool(
176                           "Want to proceed? Y/N ", "YyNn");
177
178         if(stillvalid){
179
180         }
181         else{
182           valid = false;
183         }
184
185         listoutput(vec);
186
187
188     }
189
190
191 }
192
193
194
195 void citySort( std::vector<City> &vec){
196     short typeOfSort = -1;
197     // 1 is by name, 2 by x, 3 by y
198
199 bool valid = true;
200     while(valid){
201         std::cout << " 1) sort by Name: " << std::endl;
202         std::cout << " 2) sort by X"<< std::endl;
203         std::cout << " 3) sort by Y"<< std::endl;
204         std::cout << " 4) Quit"<< std::endl;
205        char menuChoice = input_protect<char>(" ");
206
207         if(menuChoice == '1'||menuChoice == 'N'){
208             typeOfSort = 1;
209
210         }
211         else if(menuChoice == '2'||menuChoice == 'X'){
212             typeOfSort = 2;
213
214         }
215         else if(menuChoice == '3'||menuChoice == 'Y'){
216             typeOfSort = 3;
217
218         }
219          else if(menuChoice == '4'||menuChoice == 'Q'){
220             valid = false;
221
222         }
223
224         insertion(vec,typeOfSort);
225
226         listoutput(vec);
227     }
228
229 }
230
231
232 void listoutput(std::vector<City> &cityList){
233     std::cout <<  "\n  City List: \n"  << std::endl;
234
235      for (std::vector<City>::size_type i = 0;
236      i < cityList.size(); ++i)
237     {
238         std::cout << i+1 <<":  "+ cityList[i].get_name() << " ("
239         << cityList[i].get_location().get_x() <<","
240         << cityList[i].get_location().get_y() <<")" << std::endl;
241     }
242
243     std::cout <<  "\n"  << std::endl;
244
245 }
246
247
248
249 bool insert(std::vector<City> & vec,
250             std::vector<City>::size_type before_me,
251             City new_item, std::vector<City>::size_type end)
252 {
253     bool okay = before_me < end;
254     if (okay)
255     {
256         for (std::vector<City>::size_type pos = end;
257         pos > before_me; pos--)
258         {
259             vec[pos] = City(vec[pos-1]);
260         }
261         vec[before_me] = new_item;
262     }
263     return okay;
264 }
265
266
267
268 void insertion(std::vector<City> & vec, short typeOfSort)
269 {
270     std::vector<City>::size_type dest;
271     Point p;
272     City holder("City", p); //should fix the city warnings
273     if(typeOfSort == 1){
274         for (std::vector<City>::size_type next = 1;
```

```
275         next < vec.size(); ++next)
276         {
277             holder = City(vec[next]);
278             dest = next;
279             while (dest > 0 &&          // look amongst the already sorte
280                    vec[dest-1].get_name() > holder.get_name()) // for where
281                    //change this gator to change ascending order
282             {
283                 --dest;
284             }
285             if (dest != next)              // not already in place?
286             {
287                 insert(vec, dest, holder, next);    // insert 'im in front
288             }
289         }
290     }
291     else if (typeOfSort == 2){
292         for (std::vector<City>::size_type next = 1;
293         next < vec.size(); ++next)
294         {
295             holder = vec[next];
296             dest = next;
297             while (dest > 0 && // look amongst the already sorted
298                    vec[dest-1].get_location().get_x() < holder.get_locatio
299                    // for where the new guy goes (in front)
300                    //change this gator to change ascending order
301             {
302                 --dest;
303             }
304             if (dest != next)              // not already in place?
305             {
306                 insert(vec, dest, holder, next);    // insert 'im in front
307             }
308         }
309
310
311     }
312     else if (typeOfSort == 3){
313         for (std::vector<City>::size_type next = 1; next < vec.size(); ++ne
314         {
315             holder = vec[next];
316             dest = next;
317             while (dest > 0 &&          // look amongst the already sorte
318                    vec[dest-1].get_location().get_y() < holder.get_locatio
319                    // for where the new guy goes (in front)
320                    //change this gator to change ascending order
321             {
322                 --dest;
323             }
324             if (dest != next)              // not already in place?
325             {
326                 insert(vec, dest, holder, next);    // insert 'im in front
327             }
328         }
329
330
331     }
332     return;
333 }
```

```
j_pec2@ares:~/JPMainDir/CSC122/Port1/CityMapProject$ show-code
j_pec2@ares:~/JPMainDir/CSC122/Port1/CityMapProject$ show-code city.h
```

city.h:

```
 1  #ifndef CITY_H_INC
 2  #define CITY_H_INC
 3
 4  #include <iostream>
 5
 6  #include <string>
 7  #include "city.h"
 8
 9  //City Header file
10
11
12  class City
13      {
14          Point location;
15          std::string name;
16      public:
17
18          //getters
19          double distance(const City & other) const
20          {
21              return location.distance(other.location);
22          }
23
24          Point get_location(void) const
25          {
26              return location;
27          }
28
29          std::string get_name(void) const{
30              return name;
31          }
32
33          //setters
34          void set_location(Point &p)
35          {
36             location.set_x(p.get_x());
37             location.set_y(p.get_y());
38
39          }
40          /*
41          void set_location(double x, double y)
42          {
```

```
43              location.set_x(x);
44              location.set_y(y);
45
46          }
47          */
48
49          void set_name(std::string input){
50              name = input;
51
52          }
53
54
55
56
57      // other methods and constructors here
58
59
60
61
62      //constructors
63      City(void)
64          :
65            location{},
66            name{"City"}
67
68      {
69
70      }
71
72      City(const City & cityObj)
73          :
74            location{cityObj.location},
75            name{cityObj.name}
76
77      {
78
79      }
80
81      City(std::string nameIN, Point & p)
82          : City{}
83      {
84         set_location(p);
85         set_name(nameIN);
86
87
88      }
89
90      City & operator=(const City & c) = default;
91      /*
92       City(std::string nameIN, double x, double y)
93           : City{}
94      {
95         set_location(x,y);
96         set_name(nameIN);
97
98
99          }
100         */
101
102         };
103
104
105
106
107
108
109     #endif
```

```
j_pec2@ares:~/JPMainDir/CSC122/Port1/CityMapProject$ show-code Input_prot2
Unknown class/file type!  Please have your teacher request a group update...
j_pec2@ares:~/JPMainDir/CSC122/Port1/CityMapProject$ show-code Input_prot2.h
```

Input_prot2.h:

```
1   /*
2       Header file for Input_prot2 //can split this into
3       implementation and interfaces, but I think inline functions can work to
4   */
5   #pragma once //for most cases
6   #include <iostream>
7   #include <limits>
8   #include <string>
9   #include <vector>
10
11  const bool USING_MAX = 0;
12  const bool USING_MIN = 1;
13
14  //error place for edit
15  std::string errorInvalidChoice = "Error! Your choice is not valid.\n";
16  std::string errorCantRead = "Error! Cannot read input.\n";
17
18  void inline quitOrNot(bool &valid){ //Tried to make this work but its a has
19      std::cout << "Do you want to retry your entry or quit this entry?"
20              <<", type Q to quit, any other input to proceed: " ;
21
22          char in;
23          std::cin >> in;
24          if (std::cin.peek() == '\n'){
25              std::cin.ignore();
26          }
27
28          if(in == 'Q'){
29              valid = false;
30          }
31
32
33  }
```

```

template<typename inputType>
bool inline searchListMatch(inputType returnValue
                            ,const std::vector<inputType> &vec){

typedef typename std::vector<inputType>::size_type vecPos;
   bool match = false;
    for(vecPos i = 0; i < vec.size();i++){
            if(returnValue == vec[i]){
            return match = true;
            }
    }

    return match;

}


template<typename inputType>
inputType inline input_protectChoice(std::string const &prompt
                                     ,std::vector<inputType> vec)
                                     //does not work with bools
{
    inputType returnValue;

                std::cout << prompt;
            std::cin >> returnValue;

            while (std::cin.fail()==1) {
                std::cout << errorCantRead;


                std::cin.clear();
                std::cin.ignore(
                std::numeric_limits<std::streamsize>::max(),'\n');
                std::cout << prompt;
                std::cin >> returnValue;
            }

              while (searchListMatch(returnValue,vec) == false) {
                std::cout << errorInvalidChoice;
                std::cin.clear();
                std::cin.ignore(
                std::numeric_limits<std::streamsize>::max(),'\n');
                std::cout << prompt;
                std::cin >> returnValue;
            }

return returnValue;

}

template<typename inputType>
// added templates for use for more data types,
//  and inline to experiment with different kinds of libs
inline inputType input_protect(std::string const &prompt)
{

bool valid = true;
inputType returnValue=0;


while(valid == true){
        // aborting this seems hard, tried to work on it but oh well

        std::cout << prompt;
        std::cin >> returnValue;

        while (std::cin.fail()==1) {


                std::cout << errorCantRead;

                std::cin.clear();
                std::cin.ignore(
        std::numeric_limits<std::streamsize>::max(),'\n');

                std::cout << prompt;
                std::cin >> returnValue;
        }

valid = false;
}
return returnValue;

}


template<typename inputType>
inline inputType input_protect(std::string const &prompt
                                , inputType minValue)
{
        inputType returnValue=0;
        returnValue=input_protect<inputType>(prompt);
        while (returnValue < minValue) {
                std::cout << "Value must be >= " << minValue << std::endl;
                returnValue=input_protect<inputType>(prompt);
        }
        return returnValue;
}

template<typename inputType>
inline inputType input_protect(inputType maxValue
                                ,std::string const &prompt)
{
        inputType returnValue=0;
        returnValue=input_protect<inputType>(prompt);
        while (returnValue > maxValue) {
```

```
                std::cout << "Value must be <= " << maxValue << std::endl;
                std::cin.clear();
        std::cin.ignore(std::numeric_limits<std::streamsize>::max(),'\n');
                returnValue=input_protect<inputType>(prompt);
        }
        return returnValue;
}


template<typename inputType>
inline inputType input_protect(std::string const &prompt
                               , inputType minValue
                               , inputType maxValue)
{
        inputType returnValue=0;
        returnValue=input_protect<inputType>(prompt,minValue);
        while (returnValue > maxValue) {
                std::cout << "Value must be <= " << maxValue << std::endl;
                std::cin.clear();
        std::cin.ignore(std::numeric_limits<std::streamsize>::max(),'\n');
                returnValue=input_protect<inputType>(prompt,minValue);
        }
        return returnValue;
}


template<typename inputType>
inline inputType input_protect(bool i,std::string const &prompt
                               ,inputType maxormaxValue)
{
        inputType returnValue=0;
        returnValue=input_protect<inputType>(prompt);


        if(i == USING_MAX){
        while (returnValue > maxormaxValue) {
            std::cout << "Value must be <= " << maxormaxValue << std::endl
            std::cin.clear();
            std::cin.ignore(std::numeric_limits<std::streamsize>::max(),'\
            returnValue=input_protect<inputType>(prompt);
        }

        }else if(i == USING_MIN){
        while (returnValue < maxormaxValue) {
                std::cout << "Value must be >= " << maxormaxValue << std::
                std::cin.clear();
        std::cin.ignore(std::numeric_limits<std::streamsize>::max(),'\n');
                returnValue=input_protect<inputType>(prompt);
        }



        }
```

```
          return returnValue;

}


bool inline input_protectChoiceBool(std::string const &prompt
                                    ,std::string validChars)
{
    bool returnValue;

    char input = input_protect<char>(prompt);

            while (validChars.find_first_of(input)
                   == std::string::npos) {
                std::cout << errorInvalidChoice;
                std::cin.clear();
                std::cin.ignore(
                std::numeric_limits<std::streamsize>::max(),'\n');
                input = input_protect<char>(prompt);

            }

            if(input == 'Y' || input == 'y' ){
                    // this part has to be hardcoded for now
                   returnValue = true;
            }
            else{
                returnValue = false;
            }

return returnValue;

}
```

```
j_pec2@ares:~/JPMainDir/CSC122/Port1/CityMapProject$ show-code point.h
```

point.h:

```
#ifndef POINT_CLASS_HEADER_INCLUDED
#define POINT_CLASS_HEADER_INCLUDED

#include <cmath>

// A 2D point class
class Point
{
    double x, // x coordinate of point
           y; // y coordinate of point

public:


    void Output(void) const {
```

```
16          std::cout << '(' << x << ", " << y << ')';
17          return;
18      };   // output this point
19      void Input(void){
20           char dummy;
21           std::cin >> dummy
22           >> x >> dummy >> y >> dummy;
23           return;
24
25      };          // input this point
26
27      // distance between this point and other
28      double distance(const Point & other) const{
29          return sqrt(pow(x-other.x, 2.0) +
30                      pow(other.y-y, 2.0));
31      };
32      // point in middle of this point and other
33      Point midpoint(const Point & other) const{
34           return Point((x+other.x)/2.0, (other.y+y)/2.0);
35
36
37      };
38
39      double get_x(void) const { return x; }  // accessors
40      double get_y(void) const { return y; }
41
42      void set_x(double new_x){
43          x = new_x;
44          return;
45
46      };               // mutators
47      void set_y(double new_y){
48           y = new_y;
49          return;
50
51      };
52
53      //constructors
54      Point(void)
55          :
56          x{0},
57          y{0}
58      {
59
60      }
61
62      Point(const Point & p)
63           :
64            x {p.x},
65            y {p.y}
66      {
67
68      }
69
```

```
70       Point(double new_x, double new_y)
71          :Point{}
72          {
73               set_x(new_x);
74               set_y(new_y);
75
76          }
77
78
79
80      ////////////Unused
81      Point flip_x(void) const; // new point is this one flipped
82      Point flip_y(void) const; // about specified axis
83
84      Point shift_x(double move_by) const;  // new point is this one
85      Point shift_y(double move_by) const;  // shifted move_by in the
86                                            // specified direction
87
88      //   Point(void){ x = y = 0.0;};
89     // Point(double new_x, double new_y){ set_x(new_x);set_y(new_y);};
90    //  Point(const Point & p) { x = p.x; y = p.y;};
91
92      ////////////////////////////
93
94      Point & operator=(const Point & p) = default;
95  };
96
97  #endif
j_pec2@ares:~/JPMainDir/CSC122/Port1/CityMapProject$ show-copwdscrCPP CounterLabDr

pwdcat MAPDshow-codCPP CPP MAPDriverJP.cpp point.h
APDriverJP.cpp point.h

APDriverJP.cpp point.h

APDriverJP.cpp point.h

MAPDriverJP.cpp point.h




MAPDriverJP.cpp***


j_pec2@ares:~/JPMainDir/CSC122/Port1/CityMapProject$ ./MAPDriverJP.out
 1) Enter city Information
 2) calculate Distance between two cities
 3) Print All cities
 4) Sort the cities
 5) Quit
 1
City Name: Chicago
(5,7)
```

```
Want to proceed? Y/N y

  City List:

1:  Chicago (5,7)


City Name: Los Vegas
(-2,55)
Want to proceed? Y/N y

  City List:

1:  Chicago (5,7)
2:  Los Vegas (-2,55)


City Name: New York
(3,555)
Want to proceed? Y/N n

  City List:

1:  Chicago (5,7)
2:  Los Vegas (-2,55)
3:  New York (3,555)


 1) Enter city Information
 2) calculate Distance between two cities
 3) Print All cities
 4) Sort the cities
 5) Quit
 2

  City List:

1:  Chicago (5,7)
2:  Los Vegas (-2,55)
3:  New York (3,555)


Select 2 indexes for 2 Citys:
Index 1: 1
Index 2: 1
Cant pick the same city twice: Index 1: 1
Index 2: 2
Distance between Chicago and Los Vegas is: 48.5077
 1) Enter city Information
 2) calculate Distance between two cities
 3) Print All cities
 4) Sort the cities
 5) Quit
 3

  City List:

1:  Chicago (5,7)
2:  Los Vegas (-2,55)
3:  New York (3,555)


 1) Enter city Information
 2) calculate Distance between two cities
 3) Print All cities
 4) Sort the cities
 5) Quit
 4
 1) sort by Name:
 2) sort by X
 3) sort by Y
 4) Quit
 1

  City List:

1:  Chicago (5,7)
2:  Los Vegas (-2,55)
3:  New York (3,555)


 1) sort by Name:
 2) sort by X
 3) sort by Y
 4) Quit
 2

  City List:

1:  Chicago (5,7)
2:  New York (3,555)
3:  Los Vegas (-2,55)


 1) sort by Name:
 2) sort by X
 3) sort by Y
 4) Quit
 3

  City List:

1:  New York (3,555)
2:  Los Vegas (-2,55)
3:  Chicago (5,7)


 1) sort by Name:
```

```
 2) sort by X
 3) sort by Y
 4) Quit
 4

  City List:

1:  New York (3,555)
2:  Los Vegas (-2,55)
3:  Chicago (5,7)


 1) Enter city Information
 2) calculate Distance between two cities
 3) Print All cities
 4) Sort the cities
 5) Quit
 1
City Name: Night City
(1,1)
Want to proceed? Y/N n

  City List:

1:  New York (3,555)
2:  Los Vegas (-2,55)
3:  Chicago (5,7)
4:  Night City (1,1)


 1) Enter city Information
 2) calculate Distance between two cities
 3) Print All cities
 4) Sort the cities
 5) Quit
 2

  City List:

1:  New York (3,555)
2:  Los Vegas (-2,55)
3:  Chicago (5,7)
4:  Night City (1,1)


Select 2 indexes for 2 Citys:
Index 1: -3
Value must be >= 1
Index 1: 3
Index 2: 4
Distance between Chicago and Night City is: 7.2111
 1) Enter city Information
 2) calculate Distance between two cities
 3) Print All cities
 4) Sort the cities
 5) Quit
 5
j_pec2@ares:~/JPMainDir/CSC122/Port1/CityMapProject$ exit
exit

Script done on 2022-09-27 15:14:43-05:00 [COMMAND_EXIT_CODE="0"]
```