```
Script started on Sun 26 Feb 2017 11:34:22 AM CST
\033]0;b_pepa@mars:~/CSC122/map_prog\007[b_pepa@mars map_prog]$ pwd
/home/students/b_pepa/CSC122/map_prog
\033]0;b_pepa@mars:~/CSC122/map_prog\007[b_pepa@mars map_prog]$ cat map.info
Brandon Pepa
CSC122-001
Let's MAP it out...
Project

(level 4)
(level 2)
        Add a third class to manage the city list

**(level 6)**

Description:
        This program allows you to create cities and add and delete them
from a list in order to find the distances between two cities. It is user
friendly enough in order for most people to be able to use it.
\033]0;b_pepa@mars:~/CSC122/map_prog\007[b_pepa@mars map_prog]$ cat map.cpp
#include <iostream> //cout/cin
#include <cstring> //to manipulate strings and compare
#include <cmath> //distance formula
using namespace std;

class Point
{
   double x, // x coordinate of point
          y; // y coordinate of point
public:
   Point(void);
   Point(double new_x, double new_y);
   Point(const Point & p);

   void Output(void) const;
   void Input(void);

   //calculates the distance between this point and other
   double distance(const Point & other) const;
   //Returns the point that is in the middle of this point and other
   Point midpoint(const Point & other) const;

   double get_x(void) const { return x; }  // accessors
   double get_y(void) const { return y; }

   void set_x(double new_x);            //mutators
   void set_y(double new_y);
};

const size_t MAX_CITY_NAME = 50;
class City
{
   Point location;
   char name[MAX_CITY_NAME];
public:
   City(void);
   City(Point new_loc, char new_name[MAX_CITY_NAME]);
   City(const City & p);

   void Output(void) const;
   void Input(void);

   // mutators
   void set_loc(Point new_loc) { location = new_loc; return; }
   void set_name(char new_name[MAX_CITY_NAME])
   {
      strcpy(name, new_name); return;
   }

   double distance(const City & other) const
   {
      return location.distance(other.location);
   }

   // accessors
   Point get_location(void) const { return location; }
   void get_name(char p[MAX_CITY_NAME]) const { strcpy(p,name); return; }


};

const size_t MAX_CITY_LIST = 15;
class CityList
{
   City list[MAX_CITY_LIST];
   size_t current;
public:
   CityList(void) : current(0) {}

   bool full(void) const { return current == MAX_CITY_LIST; }
   bool empty(void) const { return current == 0; }

   City get(size_t index) const { return list[index]; }
   void print(void) const;

   size_t get_num_cities(void) const { return current; } // accessor

   void set(size_t index, const City & new_city);  // mutators
   bool add(const City & new_city);
   void del(size_t index);

};

void print_menu(void);


int main(void)
{
   // Menu choice variables
   char choice,
        c;

   //The main city list for the program
   CityList city_list;

   City temp;

   do
   {
      print_menu();
      cin >> choice;
      cin.ignore(INT_MAX,'\n');
      switch(choice)
      {
         case '1': case 'E': case 'e':
            temp.Input();
            if(!city_list.add(temp))
            {
```

```
            do
            {
               // Character for selecting "menu" choice
               cout << "Your city list is full." << endl
                    << "Would you like to Overwrite a city" << endl
                    << "or Discard this entry? ";
               cin >> c;
            } while( c != 'o' && c != 'O' &&
                     c != 'd' && c != 'D' );

            //If we need to overwrite a city
            if( c == 'o' || c == 'O' )
            {
               size_t position;
               city_list.print();
               cout << "\nWhich position would you like to overwrite? ";
               cin >> position;
               city_list.set(position - 1 ,temp);
            }
         }
         break;
      case '2': case 'D': case 'd':
         if( city_list.get_num_cities() == 2 )
         {
            char name1[MAX_CITY_NAME],
                 name2[MAX_CITY_NAME];

            city_list.get(0).get_name(name1);
            city_list.get(1).get_name(name2);

            // city_list.get returns the city then calls the get name
            cout << "The distance between " << name1
                 << " and "                 << name2
                 << " is " << city_list.get(0).distance( city_list.get(1) )
                 << endl;
         }
         else if( city_list.get_num_cities() > 2)
         {
            size_t city1,
                   city2;

            char name1[MAX_CITY_NAME],
                 name2[MAX_CITY_NAME];

            do
            {
            system("clear");
            city_list.print();
            cout << "Enter the positions of two cities ";
            cin >> city1 >> city2;
            } while(city1 == city2 || city1 < 1 || city2 < 1 ||
                    city1 > city_list.get_num_cities() ||
                    city2 > city_list.get_num_cities());

            // I know this is bad practice to subtract from unsigned
            // but I already protect it above...
            --city1;
            --city2;

            city_list.get(city1).get_name(name1);
            city_list.get(city2).get_name(name2);

            cout << "The distance between " << name1
                 << " and "                 << name2
                 << " is "
                 << city_list.get(city1).distance( city_list.get(city2) )
                 << endl;
            }
            else
            {
               cout << "You have not entered enough cities yet." << endl;
            }

            break;
         case '3': case 'P': case 'p':
            city_list.print();
            break;
         case '4': case 'q': case 'Q':
            break;
         default:
            cout << "Please enter a valid menu choice" << endl;
            break;

      }
     // loop until the user enters 4, q, or Q
   } while( choice != '4' && choice != 'q' && choice != 'Q' );

return 0;
}


void print_menu(void)
{
   cout << "1) Enter city information" << endl
        << "2) find Distance between two cities" << endl
        << "3) Print all cities" << endl
        << "4) Quit" << endl;
}

void Point :: Input(void)
{
   char dummy;
   cin >> dummy >> x >> dummy >> y >> dummy;
   return;
}

void Point :: Output(void) const
{
   cout << '(' << x << ", " << y << ')';
   return;
}

double Point :: distance(const Point & other) const
{
   return sqrt(pow(x-other.x,2.0) +
               pow(y-other.y,2.0));
}

Point Point :: midpoint(const Point & other) const
{
   return Point((x+other.x)/2.0, (other.y+y)/2.0);
}

void Point :: set_y(double new_y)
{
   y = new_y;
   return;
}
```

```
void Point :: set_x(double new_x)
{
   x = new_x;
   return;
}

Point :: Point(const Point & p) : x(p.x), y(p.y)
{
}

Point :: Point(void) : x(0), y(0)
{
}

Point :: Point(double new_x, double new_y) : x(), y()
{
   set_x(new_x);
   set_y(new_y);
}

City :: City(void) : location(), name()
{
}

City :: City(Point new_loc, char new_name[MAX_CITY_NAME]) : location(), name()
{
   location = new_loc;
   strcpy(name, new_name);
}

City :: City(const City & p) : location(), name()
{
   location = p.location;
   strcpy(name, p.name);
}

void City :: Output(void) const
{
   cout << name << "\t\t";
   location.Output();
   return;
}

void City :: Input(void)
{
   cout << "Enter the name of the city: ";
   cin.getline(name, INT_MAX);
   cout << "Enter the location of the city: ";
   location.Input();
   return;
}

void CityList :: print(void) const
{
   cout << "   City Name\t|\tLocation" << endl;
   for(size_t i = 0; i < current; ++i)
   {
      cout << i+1 << ") ";
      list[i].Output();
      cout << endl;
   }
   cout << endl;
   return;
}

void CityList :: set(size_t index, const City & new_city)
{
   list[index] = new_city;
   return;
}

bool CityList :: add(const City & new_city)
{
   if(!full())
   {
      list[current] = new_city;
      ++current;
      return true;
   }
   return false;
}

void CityList :: del(size_t index)
{
   for(size_t i = index; i < current; ++i)
   {
      list[i] = list[i+1];
   }
   --current;
   return;
}
```

```
\033]0;b_pepa@mars:~/CSC122/map_prog\007[b_pepa@mars map_prog]$ cat\033[K\033[K\033[K\033[KCPP
map.cpp\033[K\033[K\033[K\033[K\033[K
map.cpp***


\033]0;b_pepa@mars:~/CSC122/map_prog\007[b_pepa@mars map_prog]$ ./map.out\033[K
1) Enter city information
2) find Distance between two cities
3) Print all cities
4) Quit
1
Enter the name of the city: Chicago
Enter the location of the city: (0,3)
1) Enter city information
2) find Distance between two cities
3) Print all cities
4) Quit
2
```

```
You have not entered enough cities yet.
1) Enter city information
2) find Distance between two cities
3) Print all cities
4) Quit
1
Enter the name of the city: Palatine
Enter the location of the city: (3.6
4,3)
1) Enter city information
2) find Distance between two cities
3) Print all cities
4) Quit
3
   City Name    |       Location
1) Chicago              (0, 3)
2) Palatine             (3.6, 0.3)

1) Enter city information
2) find Distance between two cities
3) Print all cities
4) Quit
2
The distance between Chicago and Palatine is 4.5
1) Enter city information
2) find Distance between two cities
3) Print all cities
4) Quit
1
Enter the name of the city: Elgin
Enter the location of the city: (6.5,9.2)
1) Enter city information
2) find Distance between two cities
3) Print all cities
4) Quit
2
\033[H\033[2J   City Name       |       Location
1) Chicago              (0, 3)
2) Palatine             (3.6, 0.3)
3) Elgin                (6.5, 9.2)

Enter the positions of two cities 1 3
The distance between Chicago and Elgin is 8.98276
1) Enter city information
2) find Distance between two cities
3) Print all cities
4) Quit
3
   City Name    |       Location
1) Chicago              (0, 3)
2) Palatine             (3.6, 0.3)
3) Elgin                (6.5, 9.2)

1) Enter city information
2) find Distance between two cities
3) Print all cities
4) Quit
4
\033]0;b_pepa@mars:~/CSC122/map_prog\007[b_pepa@mars map_prog]$ cat\033[K\033[K033[K
\033[K033[K\007exit
exit

Script done on Sun 26 Feb 2017 11:37:15 AM CST
```