```
Script started on 2024-09-15 13:17:06-05:00 [TERM="xterm-256color" TTY="/dev/pts/3"
e_prevost@ares:~/Portfolio_1/Project 1$ pwd
/home/students/e_prevost/Portfolio_1/Project 1
e_prevost@ares:~/Portfolio_1/Project 1$ cat citydist.info
****************

Robert Prevost

CSC 122 W01



City Distance Project



Takes in multiple inputs of cities and their locations and allows user to

output the distance of those cities.



Base Level: Level 4

Third Class Added: +2 Level

Total Level: Level 6

****************e_prevost@ares:~/Portfolio_1/Project 1$ show-code cdriver_3.cpp


driver_3.cpp:
```

```cpp
#include <iostream>
#include <cstring>
#include <string>
#include <limits>
#include "point.h"

const size_t MAX_CITY_NAME = 50;
const size_t MAX_CITY_LIST = 10;

class City {
private:
    Point location;
    char name[MAX_CITY_NAME];

public:
    City() : location(), name{} {}
    City(const Point& location_prov) : location(location_prov), name{} {}

    void setName(const std::string& newName) {
        size_t i;
        for (i = 0; i < MAX_CITY_NAME - 1 && i < newName.length(); ++i) {
            name[i] = newName[i];
        }
        name[i] = '\0';  // Null terminator for string
    }

    std::string getName() const { return std::string(name); }

    void setLocation(const Point& newLocation) { location = newLocation; }
    Point getLocation() const { return location; }

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

class CityList {
private:
    City cities[MAX_CITY_LIST];
    size_t count;

public:
    CityList() : count(0) {}

    bool isFull() const { return count == MAX_CITY_LIST; }
    bool isEmpty() const { return count == 0; }
    size_t getCount() const { return count; }

    bool addCity(const City& newCity) {
        if (isFull()) return false;
        cities[count++] = newCity;
        return true;
    }

    City& getCity(size_t index) {
        if (index >= count) return cities[0];
        return cities[index];
    }

    void printAllCities() const {
        for (size_t i = 0; i < count; ++i) {
            std::cout << i+1 << ". " << cities[i].getName() << " at ";
            cities[i].getLocation().Output();
            std::cout << std::endl;
        }
    }
};

void displayMenu();
void enterCityInformation(CityList& cityList);
void calculateDistance(CityList& cityList);
void printAllCities(CityList& cityList);

int main() {
```

```
    CityList cityList;
    char choice;

    do {
        displayMenu();
        std::cin >> choice;

        switch (choice) {
            case '1':
            case 'E':
            case 'e':
                enterCityInformation(cityList);
                break;
            case '2':
            case 'C':
            case 'c':
                calculateDistance(cityList);
                break;
            case '3':
            case 'P':
            case 'p':
                printAllCities(cityList);
                break;
            case '4':
            case 'Q':
            case 'q':
                std::cout << "Goodbye!" << std::endl;
                break;
            default:
                std::cout << "Invalid choice. Please try again."
                << std::endl;
        }
    } while (choice != '4' && choice != 'Q' && choice != 'q');

    return 0;
}

void displayMenu() {
    std::cout << "\n1) Enter city Information" << std::endl;
    std::cout << "2) Calculate Distance between two cities" << std::endl;
    std::cout << "3) Print All cities" << std::endl;
    std::cout << "4) Quit" << std::endl;
    std::cout << "Enter your choice: ";
}

void enterCityInformation(CityList& cityList) {
    if (cityList.isFull()) {
        std::cout << "The city list is full. Cannot add more cities."
        << std::endl;
        return;
    }

    double x = 0.0, y = 0.0;
    std::string cityName;

    std::cout << "Enter coordinates of City (x,y): ";
    Point new_city_point;
    new_city_point.Input();

    City newCity(new_city_point);

    std::cout << "Enter name of City: ";
    std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
    std::getline(std::cin, cityName);

    newCity.setName(cityName);

    if (cityList.addCity(newCity)) {
        std::cout << "City added successfully!" << std::endl;
    } else {
        std::cout << "Failed to add city. The list might be full."
        << std::endl;
    }
}

void calculateDistance(CityList& cityList) {
    if(cityList.getCount() < 2){
        std::cout << "Please Add More Cities!\n";
    }
    else{
        cityList.printAllCities();
        size_t index1, index2;
        std::cout << "Enter # of First City: ";
        std::cin >> index1;
        std::cout << "Enter # of Second City: ";
        std::cin >> index2;
        index1--;
        index2--;
        City& firstCity = cityList.getCity(index1);
        City& secondCity = cityList.getCity(index2);
        double distance = firstCity.distance(secondCity);
        std::cout << "These cities are " << distance << " units apart.\n";
    }
}

void printAllCities(CityList& cityList) {
    cityList.printAllCities();
}
```

```
e_prevost@ares:~/Portfolio_1/Project 1$ show-code point.h
```

point.h:

```
#ifndef POINT_CLASS_HEADER_INCLUDED
#define POINT_CLASS_HEADER_INCLUDED

// A 2D point class
```

```
    5  class Point
    6  {
    7      double x, // x coordinate of point
    8             y; // y coordinate of point
    9
   10  public:
   11      Point(void);
   12      Point(double new_x, double new_y);
   13      Point(const Point & p);
   14
   15      void Output(void) const;   // output this point
   16      void Input(void);          // input this point
   17
   18      // distance between this point and other
   19      double distance(const Point & other) const;
   20      // point in middle of this point and other
   21      Point midpoint(const Point & other) const;
   22
   23      double get_x(void) const { return x; }  // accessors
   24      double get_y(void) const { return y; }
   25
   26      void set_x(double new_x);               // mutators
   27      void set_y(double new_y);
   28
   29      Point flip_x(void) const; // new point is this one flipped
   30      Point flip_y(void) const; // about specified axis
   31
   32      Point shift_x(double move_by) const;  // new point is this one
   33      Point shift_y(double move_by) const;  // shifted move_by in the
   34                                            // specified direction
   35  };
   36
   37  #endif
```

```
e_prevost@ares:~/Portfolio_1/Project 1$ show-code point.cpp
```

point.cpp:

```
    1  #include "point.h"
    2
    3  #include <iostream>
    4  #include <cmath>
    5
    6  using namespace std;
    7
    8  // read standard 2D point notation (x,y) -- ignore
    9  // window dressing
   10  void Point::Input(void)
   11  {
   12      char dummy;
   13      cin >> dummy >> x >> dummy >> y >> dummy;
   14      return;
   15  }
   16
   17  // output standard 2D point notation (x,y)
   18  void Point::Output(void) const
   19  {
   20      cout << '(' << x << ", " << y << ')';
   21      return;
   22  }
   23
   24  // calculate distance between two 2D points --
   25  // the one that called us and the argument
   26  double Point::distance(const Point & other) const
   27  {
   28      return sqrt(pow(x-other.x, 2.0) +
   29                  pow(other.y-y, 2.0));
   30  }
   31
   32  // calculate midpoint between two 2D points --
   33  // the one that called us and the argument
   34  Point Point::midpoint(const Point & other) const
   35  {
   36      return Point((x+other.x)/2.0, (other.y+y)/2.0);
   37  }
   38
   39  // set coordinates to programmer-specified values
   40  void Point::set_x(double new_x)
   41  {
   42      x = new_x;        // no error checking since anything is legal
   43      return;
   44  }
   45
   46  // set coordinates to programmer-specified values
   47  void Point::set_y(double new_y)
   48  {
   49      y = new_y;        // no error checking since anything is legal
   50      return;
   51  }
   52
   53  // construct Point as copy of previous point
   54  Point::Point(const Point & p)
   55  {
   56      x = p.x;
   57      y = p.y;
   58  }
   59
   60  // construct Point by default -- no values specified
   61  Point::Point(void)
   62  {
   63      x = y = 0.0;
   64  }
   65
   66  // construct Point given initial x,y values
   67  Point::Point(double new_x, double new_y)
   68  {
   69      set_x(new_x);
```

```
    70      set_y(new_y);
    71  }
    72
    73  // creates a point flipped about the x axis from us
    74  Point Point::flip_x(void) const
    75  {
    76      return Point(x,-y);
    77  }
    78
    79  // creates a point flipped about the y axis from us
    80  Point Point::flip_y(void) const
    81  {
    82      return Point(-x,y);
    83  }
    84
    85  // creates a point shifted along the x axis from us
    86  Point Point::shift_x(double move_by) const
    87  {
    88      return Point(x+move_by,y);
    89  }
    90
    91  // creates a point shifted along the y axis from us
    92  Point Point::shift_y(double move_by) const
    93  {
    94      return Point(x,y+move_by);
    95  }
e_prevost@ares:~/Portfolio_1/Project 1$ CPP driver_3 point
driver_3.cpp***
point.cpp...
driver_3.cpp: In member function ‘void
City::setLocation(const Point&)’:
driver_3.cpp:29:61: warning:
implicitly-declared ‘constexpr Point&
Point::operator=(const Point&)’ is deprecated
[-Wdeprecated-copy]
   29 |     void setLocation(const Point& newLocation) { location =
   newLocation; }
      |
      ^~~~~~~~~~
In file included from driver_3.cpp:5:
point.h:13:5: note:
because ‘Point’ has user-provided
‘Point::Point(const Point&)’
   13 |     Point(const Point & p);
      |     ^~~~~
driver_3.cpp: In function ‘void
enterCityInformation(CityList&)’:
driver_3.cpp:127:12: warning: unused
variable ‘x’ [-Wunused-variable]
  127 |     double x = 0.0, y = 0.0;
      |            ^
driver_3.cpp:127:21: warning: unused
variable ‘y’ [-Wunused-variable]
  127 |     double x = 0.0, y = 0.0;
      |                     ^
point.cpp: In copy constructor
‘Point::Point(const Point&)’:
point.cpp:54:1: warning:
‘Point::x’ should be initialized in the member
initialization list [-Weffc++]
   54 | Point::Point(const Point & p)
      | ^~~~~
point.cpp:54:1: warning:
‘Point::y’ should be initialized in the member
initialization list [-Weffc++]
point.cpp: In constructor
‘Point::Point()’:
point.cpp:61:1: warning:
‘Point::x’ should be initialized in the member
initialization list [-Weffc++]
   61 | Point::Point(void)
      | ^~~~~
point.cpp:61:1: warning:
‘Point::y’ should be initialized in the member
initialization list [-Weffc++]
point.cpp: In constructor ‘Point::Point(double,
double)’:
point.cpp:67:1: warning:
‘Point::x’ should be initialized in the member
initialization list [-Weffc++]
   67 | Point::Point(double new_x, double new_y)
      | ^~~~~
point.cpp:67:1: warning:
‘Point::y’ should be initialized in the member
initialization list [-Weffc++]


e_prevost@ares:~/Portfolio_1/Project 1$ ./driver_3.out

1) Enter city Information
2) Calculate Distance between two cities
3) Print All cities
4) Quit
Enter your choice: 1
Enter coordinates of City (x,y): (1000,2000)
Enter name of City: Banana
City added successfully!

1) Enter city Information
2) Calculate Distance between two cities
3) Print All cities
4) Quit
Enter your choice: 1
Enter coordinates of City (x,y): (2000,40000)
Enter name of City: Potato
City added successfully!

1) Enter city Information
```

```
2) Calculate Distance between two cities
3) Print All cities
4) Quit
Enter your choice: 1
Enter coordinates of City (x,y): (0,0)
Enter name of City: Chicago
City added successfully!

1) Enter city Information
2) Calculate Distance between two cities
3) Print All cities
4) Quit
Enter your choice: 3
1. Banana at (1000, 2000)
2. Potato at (2000, 40000)
3. Chicago at (0, 0)

1) Enter city Information
2) Calculate Distance between two cities
3) Print All cities
4) Quit
Enter your choice: 2
1. Banana at (1000, 2000)
2. Potato at (2000, 40000)
3. Chicago at (0, 0)
Enter # of First City: 1
Enter # of Second City: 3
These cities are 2236.07 units apart.

1) Enter city Information
2) Calculate Distance between two cities
3) Print All cities
4) Quit
Enter your choice: 2
1. Banana at (1000, 2000)
2. Potato at (2000, 40000)
3. Chicago at (0, 0)
Enter # of First City: 2
Enter # of Second City: 3
These cities are 40050 units apart.

1) Enter city Information
2) Calculate Distance between two cities
3) Print All cities
4) Quit
Enter your choice: 4
Goodbye!
e_prevost@ares:~/Portfolio_1/Project 1$ ls
citydist.info  driver_3.cpp  driver_3.out  point.cpp  point.h  typescript
e_prevost@ares:~/Portfolio_1/Project 1$ exit
exit

Script done on 2024-09-15 13:19:53-05:00 [COMMAND_EXIT_CODE="0"]
```