

CS 221 C++ Basics Evaluation Assignment

Lily Larsen

December 2, 2022

1.

<https://github.com/Id405/cs-221-eval-assignments/blob/main/eval-7/1.cpp>

```
1  #include <cmath>
2  #include <tuple>
3  #include <vector>
4  using namespace std;
5
6  class Car {
7  private:
8      double x;
9      double y;
10     double milesPerGallon;
11     double fuelTankCapacityGallons;
12     double fuelGallons;
13
14 public:
15     Car(double x, double y, float milesPerGallon, double fuelTankCapacityGallons,
16         double fuelGallons)
17         : x(x), y(y), milesPerGallon(milesPerGallon),
18           fuelTankCapacityGallons(fuelTankCapacityGallons) {}
19     bool moveTo(double destinationX, double destinationY) {
20         double gallonCost =
21             hypot(x - destinationX, y - destinationY) * milesPerGallon;
22         if (gallonCost > fuelGallons) {
23             return false;
24         }
25
26         fuelGallons -= gallonCost;
27         x = destinationX;
28         y = destinationY;
29         return true;
30     }
31     double refillTank() {
32         double refilledGallons = fuelTankCapacityGallons - fuelGallons;
33         return refilledGallons;
34     }
35 };
36
37 vector<Car> moveToPoint(vector<Car> cars, double destinationX,
38     double destinationY) {
39     vector<Car> result;
40     for (Car car : cars) {
41         if (car.moveTo(destinationX, destinationY)) {
42             result.push_back(car);
43         }
44     }
45
46     return result; // Return array with size and a lot of other helpful things
47                   // like god (the C++ standard library) intended us to do. This
48                   // helpful construct is called a vector. I know its not an
49                   // array. I know its not an array. I
50 }
51
52 class GasStation {
53 private:
```

```

54     double x;
55     double y;
56     double pricePerGallon;
57
58 public:
59     GasStation(double x, double y, double pricePerGallon)
60         : x(x), y(y), pricePerGallon(pricePerGallon) {}
61     double getX() { return x; }
62     double getY() { return y; }
63     double getPricePerGallon() { return pricePerGallon; }
64 };
65
66 tuple<vector<Car>, vector<double>>
67 moveToPointGasStations(vector<Car> cars, vector<GasStation> gasStations,
68     double destinationX, double destinationY) {
69     vector<Car> resultCar;
70     vector<double> resultCost;
71
72     for (Car car : cars) {
73         double cost = 0;
74
75         for (int i = gasStations.size(); i > 0; i--) {
76             if (car.moveTo(destinationX, destinationY)) {
77                 resultCar.push_back(car);
78                 resultCost.push_back(cost);
79                 continue;
80             }
81
82             GasStation gasStation = gasStations.at(i);
83             if (car.moveTo(gasStation.getX(), gasStation.getY())) {
84                 cost += car.refillTank() * gasStation.getPricePerGallon();
85             } else {
86                 continue;
87             }
88         }
89     }
90
91     return {resultCar, resultCost};
92 }
93
94 int main() {
95     return 0;
96 }

```

2.

<https://github.com/Id405/cs-221-eval-assignments/blob/main/eval-7/2.cpp>

```
1  #include <cmath>
2  #include <list>
3  #include <string>
4  #include <tuple>
5  #include <vector>
6  using namespace std;
7
8  enum SortingMethod { unsorted, length, value };
9
10 bool stringLessValue(string a, string b) {
11     for (int i = 0; i < a.length() && i < b.length(); i++) {
12         if (a.at(i) == b.at(i)) {
13             continue;
14         }
15         if (a.at(i) < b.at(i)) {
16             return true;
17         }
18         return false;
19     }
20
21     return false;
22 }
23
24 bool stringLessLength(string a, string b) { return a.length() < b.length(); }
25
26 class SortedList {
27 private:
28     SortingMethod sortingMethod;
29     list<string> values;
30
31 public:
32     SortingMethod getSortingMethod() const { return sortingMethod; }
33
34     void addSorted(string string) {
35         values.push_back(string);
36         sort();
37     }
38
39     void remove(string string) {
40         values.remove(string);
41     }
42
43     void removeAll() {
44         values.clear();
45     }
46
47     bool sortedByLength() const {
48         return getSortingMethod() == SortingMethod::length;
49     }
50
51     bool sortedByValue() const {
52         return getSortingMethod() == SortingMethod::value;
53     }
54 }
```

```

54
55     int length() const {
56         return values.size();
57     }
58
59     vector<string> string_values() const {
60         vector<string> result;
61
62         for (string value : values) {
63             result.push_back(value);
64         }
65
66         return result;
67     }
68
69     string at(int i) const {
70         return string_values().at(i);
71     }
72
73     void sortByLength() {
74         sortingMethod = SortingMethod::length;
75         sort();
76     }
77
78     void sortByValue() {
79         sortingMethod = SortingMethod::value;
80         sort();
81     }
82
83     void sort() {
84         if (sortingMethod == SortingMethod::unsorted) {
85             sortingMethod = SortingMethod::length; // Sort by length by default
86         }
87
88         if (sortingMethod == SortingMethod::value) {
89             values.sort(stringLessValue);
90         }
91
92         if (sortingMethod == SortingMethod::length) {
93             values.sort(stringLessLength);
94         }
95     }
96
97     SortedList() : sortingMethod(SortingMethod::unsorted) {}
98     SortedList(const SortedList &original) {
99         sortingMethod = original.getSortingMethod();
100         values = original.values;
101     }
102     SortedList(SortingMethod sortingMethod) : sortingMethod(sortingMethod) {}
103 };

```