CS 221 Assembly Basics Evaluation Assignment

Lily Larsen

December 2, 2022

1.

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

```cpp
#include <cmath>
#include <tuple>
#include <vector>
using namespace std;

class Car {
private:
  double x;
  double y;
  double milesPerGallon;
  double fuelTankCapacityGallons;
  double fuelGallons;

public:
  Car(double x, double y, float milesPerGallon, double fuelTankCapacityGallons,
      double fuelGallons)
      : x(x), y(y), milesPerGallon(milesPerGallon),
        fuelTankCapacityGallons(fuelTankCapacityGallons) {}
  bool moveTo(double destinationX, double destinationY) {
    double gallonCost =
        hypot(x - destinationX, y - destinationY) * milesPerGallon;
    if (gallonCost > fuelGallons) {
      return false;
    }

    fuelGallons -= gallonCost;
    x = destinationX;
    y = destinationY;
    return true;
  }
  double refillTank() {
    double refilledGallons = fuelTankCapacityGallons - fuelGallons;
    return refilledGallons;
  }
};

vector<Car> moveToPoint(vector<Car> cars, double destinationX,
                        double destinationY) {
  vector<Car> result;
  for (Car car : cars) {
    if (car.moveTo(destinationX, destinationY)) {
      result.push_back(car);
    }
  }

  return result; // Return array with size and a lot of other helpful things
                 // like god (the C++ standard library) intended us to do. This
                 // helpful construct is called a vector. I know its not an
                 // array. I know its not an array. I know its not an array. I
}

class GasStation {
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

```cpp
#include <cmath>
#include <list>
#include <string>
#include <tuple>
#include <vector>
using namespace std;

enum SortingMethod { unsorted, length, value };

bool stringLessValue(string a, string b) {
  for (int i = 0; i < a.length() && i < b.length(); i++) {
    if (a.at(i) == b.at(i)) {
      continue;
    }
    if (a.at(i) < b.at(i)) {
      return true;
    }
    return false;
  }

  return false;
}

bool stringLessLength(string a, string b) { return a.length() < b.length(); }

class SortedList {
private:
  SortingMethod sortingMethod;
  list<string> values;

public:
  SortingMethod getSortingMethod() const { return sortingMethod; }

  void addSorted(string string) {
    values.push_back(string);
    sort();
  }

  void remove(string string) {
    values.remove(string);
  }

  void removeAll() {
    values.clear();
  }

  bool sortedByLength() const {
    return getSortingMethod() == SortingMethod::length;
  }

  bool sortedByValue() const {
    return getSortingMethod() == SortingMethod::value;
  }
```

```
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 };
```