- Exercises

Let's test our knowledge of literals with these exercises.

WE'LL COVER THE FOLLOWING

Exercise 1

Extend the `MyDistance` class we saw in the previous lesson to support the following units:

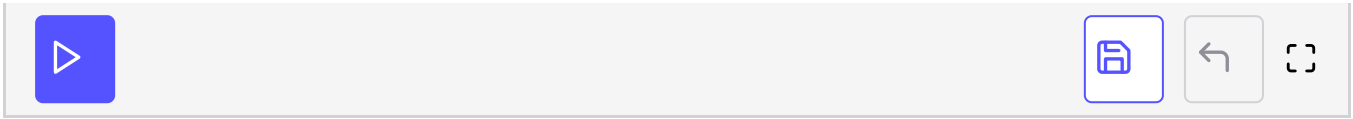
- **Feet**: 1 ft = 0.3048m
- **Mile**: 1 mi = 1609.344m

Choose good suffixes for these units.

Write an implementation below:

```
#include <iostream>
#include <ostream>

namespace Distance{
  class MyDistance{
    public:
      MyDistance(double i):m(i){}

      friend MyDistance operator +(const MyDistance& a, const MyDistance& b){
        return MyDistance(a.m + b.m);
      }
      friend MyDistance operator -(const MyDistance& a, const MyDistance& b){
        return MyDistance(a.m - b.m);
      }

      friend std::ostream& operator<< (std::ostream &out, const MyDistance& myDist){
        out << myDist.m << " m";
         return out;
      }
    private:
```

```
    double m;

  };


  namespace Unit{
    MyDistance operator "" _km(long double d){
      return MyDistance(1000*d);
    }
    MyDistance operator "" _m(long double m){
      return MyDistance(m);
    }
    MyDistance operator "" _dm(long double d){
      return MyDistance(d/10);
    }
    MyDistance operator "" _cm(long double c){
      return MyDistance(c/100);
    }
  }
}

using namespace Distance::Unit;

int main(){

  std:: cout << std::endl;

  std::cout << "1.0_km: " << 1.0_km << std::endl;
  std::cout << "1.0_m: " << 1.0_m << std::endl;
  std::cout << "1.0_dm: " << 1.0_dm << std::endl;
  std::cout << "1.0_cm: " << 1.0_cm << std::endl;


  std::cout << std::endl;
  std::cout << "1.0_km + 2.0_dm +  3.0_dm - 4.0_cm: " << 1.0_km + 2.0_dm +  3.0_dm - 4.0_cm <
  std::cout << std::endl;

}
```



Exercise 2

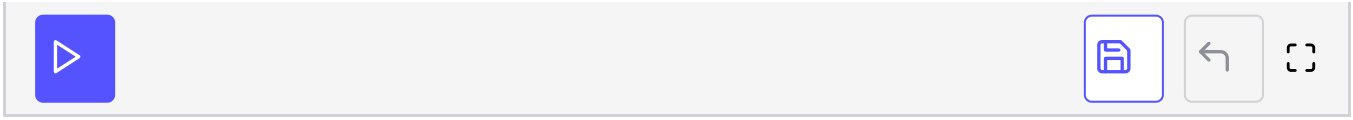
The total distance of someone's weekly drive consists of many components. Extend `MyDistance` so that we can calculate the total distance based on the following formula:

$$myDistPerWeek = 4 * work * 2 - 3 * abbrevationToWork + workout + shop$$

`work` is in `km` whereas all the others are in `m`. All of them are long doubles.

```
#include <iostream>
#include <ostream>
```

```
namespace Distance{
  class MyDistance{
    public:
      MyDistance(double i):m(i){}

      friend MyDistance operator +(const MyDistance& a, const MyDistance& b){
        return MyDistance(a.m + b.m);
      }
      friend MyDistance operator -(const MyDistance& a, const MyDistance& b){
        return MyDistance(a.m - b.m);
      }

      friend std::ostream& operator<< (std::ostream &out, const MyDistance& myDist){
        out << myDist.m << " m";
         return out;
      }
    private:
      double m;

  };

  namespace Unit{
    MyDistance operator "" _km(long double d){
      return MyDistance(1000*d);
    }
    MyDistance operator "" _m(long double m){
      return MyDistance(m);
    }
    MyDistance operator "" _dm(long double d){
      return MyDistance(d/10);
    }
    MyDistance operator "" _cm(long double c){
      return MyDistance(c/100);
    }
  }
}

using namespace Distance::Unit;

int main(){

  std:: cout << std::endl;

  std::cout << "1.0_km: " << 1.0_km << std::endl;
  std::cout << "1.0_m: " << 1.0_m << std::endl;
  std::cout << "1.0_dm: " << 1.0_dm << std::endl;
  std::cout << "1.0_cm: " << 1.0_cm << std::endl;


  std::cout << std::endl;
  std::cout << "1.0_km + 2.0_dm +  3.0_dm - 4.0_cm: " << 1.0_km + 2.0_dm +  3.0_dm - 4.0_cm <
  std::cout << std::endl;

}
```



The next lesson contains the **solution** for both exercises.

#