CSCI 320, Life Beyond Python, Spring 2022
Project 5: A templated dot product in C++

Function templates

In C++ you can create function templates that operate with variable types as parameters. For instance, the following template can be used to create a function that computes the mean of a STL (Standard Template Library) `vector` of type `T`, accumulating and returning the result in a variable of type `S`:

```
#include <vector>

template <typename S, typename T> S mean(const std::vector<T> &x)
{
  S mean = 0.0;
  for (int i = 0; i < x.size(); i++) {
    mean += x[i];
  }
  mean /= x.size();
  return mean;
}
```

If we use the range-for to iterate over the vector the code would look like this:

```
#include <vector>

template <typename S, typename T> S mean(const std::vector<T> &x)
{
  S mean = 0.0;
  for (auto &e : x) {
    mean += e;
  }
  mean /= x.size();
  return mean;
}
```

These are only templates—we need to instantiate them with specific types `S` and `T` to actually generate code:

```
#include <iostream>
#include <vector>

template <typename S, typename T> S mean(const std::vector<T> &x)
{
  S mean = 0.0;
  for (int i = 0; i < x.size(); i++) {
    mean += x[i];
  }
  mean /= x.size();
  return mean;
}

int main(int argc, char **argv)
{
  std::vector<float> x(42, 1);  // A float vector of 42 ones.
  std::vector<int> n(42, 1); // An int vector of 42 ones.
  std::cout << mean<double, float>(x) << std::endl;
  std::cout << mean<int, int>(n) << std::endl;
}
```

Project description

Recall our friend the dot product: given two vectors $(x_1, x_2, \ldots, x_n)$ and $(y_1, y_2, \ldots, y_n)$, their dot product is

$$x_1 y_1 + x_2 y_2 + \cdot + x_n y_n.$$

Starting with the code in `dot.hpp`, write a C++ function template for a function that returns the dot product of two vectors. Do not change the signature of the function template you are given. Templates are typically placed in header files since they need to be visible in order to generate the appropriate code. You can read more about the challenges of making template instantiation work here.

What to do

Create your C++ code in the file named `dot.hpp`. If you want to test it while keeping your test code in a separate file, here is how you do it. If your test code (including `main()`) is in the file `main.c`, then you can build an executable with

```
g++ -Wall -pedantic -o dot main.c
./dot
```

You will need to include your `dot.hpp` in `main()` using the preprocessor directive

```
#include "dot.hpp"
```

Your code should be inside the namespace `csci320` to avoid collisions with my code during grading. You would instantiate the function and invoke it as follows:

```
csci320::dot<double, float>(x, y);
```

Submit your file `dot.hpp` to the autograder on Gradescope for grading.