Problem 1

Write and test a function which takes

1. an array of objects of type **function<double(double)>** representing functions of type **double** → **double**,
2. its size,
3. two **double**s defining an interval $[a, b]$,
4. address of an existing **double** variable.

The function finds the maximum in the interval $[a, b]$ for all functions passed in the array and returns the one for which the maximum is the largest.

In order to find the maximum, you can go through the range $[a, b]$ with sufficiently small step (e.g., $\epsilon = 10^{-5}$) and calculate the value of the function in each of these points.

The value of the variable pointed to by the pointer passed as the last argument (pxmax) should be set equal to the value of abscissa (the 'x') at which the maximum has been reached by the returned function.

In your test program, you can use your own functions and at least one function defined by a lambada expression.

For example, the program

download *FunArrayMax.cpp*

```
#include <functional>
#include <iostream>

double f1(double x) { return  x/4; }
double f2(double x) { return -2*x; }

using D2D = std::function<double(double)>;

D2D maxfun(D2D funs[], size_t size,
           double a, double b, double* pxmax) {
    static constexpr double eps = 1e-6;
    // ...
}

int main() {
    D2D funcs[]{
        f1,
        // lambda expression here
        f2
    };
```

```
    double xmax;
    D2D pf = maxfun(funcs, 3, 0, 3, &xmax);
    std::cout << "xmax = " << xmax << "; f(xmax) = "
              << pf(xmax)  << std::endl;
}
```

for a lambda corresponding to function $f(x) = x^2$ should print

```
xmax = 3; f(xmax) = 8.99999
```
