Problem 1

Write a function

```
void mysort(int arr[], size_t size,
            std::function<bool(int,int)> cmp) {
```

which sorts (using arbitrary algorithm) the array arr. To compare elements of the array, it uses the cmp function which returns true if, and only if, the first element is considered strictly smaller than the second. Use lambdas to define the sorting criteria. The following program, after supplying missing code

download *SortLam.cpp*

```
#include <iostream>
#include <functional>  // std::function
#include <utility>     // std::swap (not indispensable)

void mysort(int arr[], size_t size,
            std::function<bool(int,int)> cmp) {
    // ...
}

// ...

int main() {
    int a[]{3, 77, 21, 19, 2, 54, 28, 91};
    size_t size = std::size(a);

    std::cout << "Normal (natural) order\n";
    mysort(a, size, /* lambda */);
    printArr(a, size);

    std::cout << "Natural order reversed\n";
    mysort(a, size, /* lambda */);
    printArr(a, size);

    std::cout << "By sum of digits, then natural\n";
    mysort(a, size, /* lambda */);
    printArr(a, size);

    int mod{};
    auto byrem = /* lambda */;
    for (int i : {3, 5, 7}) {
        mod = i;
```

```
            std::cout << "By remainder mod " << mod
                      << ", then natural reversed\n";
            mysort(a, size, byrem);
            printArr(a, size);
        }
    }
```

should print

```
Normal (natural) order
[ 2 3 19 21 28 54 77 91 ]
Natural order reversed
[ 91 77 54 28 21 19 3 2 ]
By sum of digits, then natural
[ 2 3 21 54 19 28 91 77 ]
By remainder mod 3, then natural reversed
[ 54 21 3 91 28 19 77 2 ]
By remainder mod 5, then natural reversed
[ 91 21 77 2 28 3 54 19 ]
By remainder mod 7, then natural reversed
[ 91 77 28 21 2 3 54 19 ]
```
