Problem 1

Define three function templates:

- Filtering function

  ```
  template <typename T, typename FunType>
  vector<T> filter(const vector<T>& v, FunType p);
  ```

  which takes a vector `v` and a function (a predicate) `p` taking data of the type of the single element of the vector and returning **bool**. The function **filter** returns a vector of the same type as `v` containing only those elements of `v`, for which the predicate **p** returns **true**.
- Transforming and filtering function

  ```
  template <typename T, typename FunType1, typename FunType2>
  vector<T> transfilt(vector<T>& v, FunType1 t, FunType2 p);
  ```

  which takes a vector `v`, a transforming function **t** and a predicate **p**. The vector `v` (passed by reference) is modified in such a way that its elements are replaced with the results of applying the function **t** to them. The **transfilt** function returns a vector of the same type as `v` containing only those elements of `v`, for which the predicate **p** returns **true**.
- Printing function

  ```
  template <typename T>
  void printVec(const vector<T>& v) {
  ```

  taking a vector and printing, in one line enclosed in square brackets, all its elements separated with spaces.

In invocations of **filter** and **transfilt**, function arguments should have the form of lambdas defined directly on the argument list.

If, in the following program

download *VecLam.cpp*

```
#include <cmath>
#include <iostream>
#include <functional>
#include <vector>

using std::vector;
using std::function;

template <typename T, typename FunType>
vector<T> filter(const vector<T>& v, FunType p) {
    // ...
```

```
}

template <typename T, typename FunType1, typename FunType2>
vector<T> transfilt(vector<T>& v, FunType1 t, FunType2 p) {
    // ...
}

template <typename T>
void printVec(const vector<T>& v) {
    // ...
}

int main() {
    vector<int> v{1, -3, 4, -2, 6, -8, 5};
    printVec(v);
    vector<int> r = filter(v, /* lambda_1 */);
    printVec(r);
    vector<int> s = filter(v, /* lambda_2 */);
    printVec(s);

    vector<double> w{1.5, -3.1, 4.0, -2.0, 6.3};
    printVec(w);
    double mn = -0.5, mx = 0.5;
    vector<double> d =
        transfilt(w, /* lambda_3*/, /* lambda_4 */);
    printVec(w);
    printVec(d);
}
```

lambdas are

- **lambda_1** — returns **true** for even numbers;
- **lambda_2** — returns **true** for positive numbers;
- **lambda_3** — returns sine of the argument (**std::sin** from header ***cmath***);
- **lambda_4** — returns **true** for numbers from interval $[mn, mx]$,

then it should print

```
[ 1 -3 4 -2 6 -8 5 ]
[ 4 -2 6 -8 ]
[ 1 4 6 5 ]
[ 1.5 -3.1 4 -2 6.3 ]
[ 0.997495 -0.0415807 -0.756802 -0.909297 0.0168139 ]
[ -0.0415807 0.0168139 ]
```

Note: do not use any additional functions from the Standard Library.
