Zadanie 1

Napisz funkcję

```
double neville(const double xa[], const double ya[],
                               size_t size, double x);
```

która dla danej wartości x znajduje za pomocą algorytmu Neville'a wartość wielomianu interpolacyjnego stopnia size-1 przechodzącego przez size punktów o współrzędnych przekazanych w tablicach xa i ya. Funkcja może utworzyć (dynamiczną) kopię tablicy ya — nie twórz jednak żadnych innych pomocniczych tablic ani kolekcji!

Na przykład następujący program

download *MN-Neville.cpp*

```
#include <iostream>
#include <cstring> // memcpy
double neville(const double xa[], const double ya[],
                               size_t size, double x) {
    // ...
}

int main() {
    double x[]{-1, 0, 1, 2};
    double f[]{-5,-1,-1, 1};  // x**3 - 2x**2 + x - 1
    std::cout << neville(x,f,4,3) << std::endl; // 11
}
```

powinien wydrukować 11.

Zadanie 2

Napisz funkcję

```
void polycoeff(const double x[], const double y[],
                           double c[], size_t size);
```

która oblicza i umieszcza w tablicy c współczynniki wielomianu interpolacyjnego stopnia size-1 przechodzącego przez size punktów o współrzędnych przekazanych w tablicach x i y. Skorzystaj z funkcji implementującej algorytm Neville'a. Funkcja może utworzyć (dynamiczną) kopię tablicy y — nie twórz jednak żadnych innych pomocniczych tablic lub kolekcji!

Na przykład następująca funkcja **main**:

```
int main() {
    double x[] = {-1, 0, 2,  3};
    double y[] = { 1, 1, 7, 25};  // x^3 - x + 1
    const size_t SIZE = sizeof(x)/sizeof(*x);
    double c[SIZE];

    polycoeffs(x,y,c,SIZE);
    for (int i = 0; i < SIZE; ++i)
        std::cout << c[i] << " ";
    std::cout << std::endl;
}
```

powinna wydrukować 1 -1 0 1.
