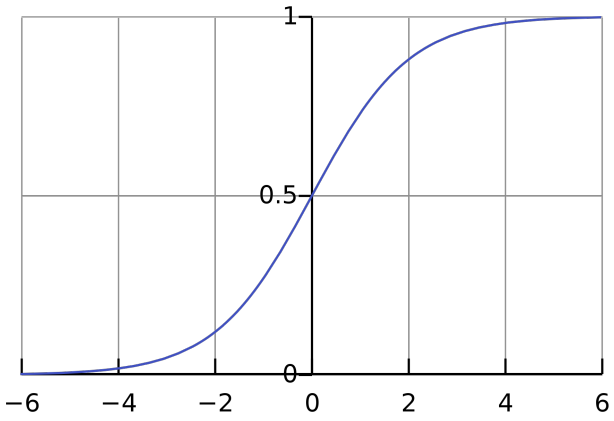
Zadanie 0 – interpolacja (1pkt)

Namnażanie się wirusów czy bakterii, symuluje się często z wykorzystaniem *funkcji logistycznej*, zdefiniowanej:

$$f(x) = \frac{L}{1 + e^{-k(x-x_0))}}, \tag{1}$$

gdzie L jest maksymalną wartością funkcji (normalizacja), k jest pochyłością krzywej, a x_0 jest położeniem środka krzywej.



Rysunek 1: Krzywa logistyczna dla $L = 1, k = 1, x_0 = 0$

Napisz funkcję **lagrint** obliczającą wartość wielomianu Lagrange'a:

$$L_f(x) = \sum_{i=0}^{n} f(x_i) \prod_{j=0 \wedge j \neq i}^{n} \frac{x - x_j}{x_i - x_j} \tag{2}$$

Wielomian ten interpoluje dowolną funkcję f. Funkcja lagrint powinna przyjmować jako argumenty:

1. wartość x
2. tablicę wartości x_i w podanych punktach (trzeba wcześniej taką wygenerować!)
3. tablicę odpowiadających im wartości $f(x_i)$ (również wygenerowana wcześniej)
4. liczbę punktów n (rozmiar obu tablic).
5. może również przyjmować argumenty do podania dalej do funkcji logistycznej

Następnie przetestuj funkcję lagrint dla funkcji logistycznej z parametrami: $L = 100, x_0 = 14, k = 2$. wypisując na ekran wartości f w równoodległych punktach x_i wraz z wartościami interpolowanymi za pomocą funkcji lagrint (wartości powinny być jednakowe).

Przykład działania:
x y interp
8 0.000614417 0.000614417
8.12 0.000781076 0.000781076

8.24 0.000992941 0.000992941
8.36 0.00126227 0.00126227
8.48 0.00160466 0.00160466
8.6 0.00203991 0.00203991
. . .

Uwaga: Nie używać żadnych algorytmów do interpolacji z bibliotek.
Podpowiedź: Funkcję eksponencjalną można znaleźć w bibliotece $< cmath >$, nazywa się *exp*. Będziesz potrzebował/a w sumie 5 tablic: znanych argumentów funkcji, znanych wartości funkcji, punktów do interpolowania, prawdziwych wartości funkcji dla interpolowanych argumentów, interpolowanych wartości funkcji dla interpolowanych argumentów.
Dla chętnych (nie punktowane): Napisz funkcję **lagrint** z użyciem szablonów funkcji. Po skończeniu narysuj wykres z punktami znanymi w jednym kolorze i interpolowanymi w innym. Jeżeli nie znasz żadnych narzędzi do rysowania to możesz wyeksportować dane do formatu .csv. Aby to zrobić, ustaw wypisywanie interesujących Cię danych oddzielonych przecinkiem, nowa linia ma oznaczać nowy zestaw danych (nowy wiersz w tablicy). Następnie odpal program i przekieruj jego wynik do pliku: **./zad0 > out.csv**

Format csv to Comma Separated Value, a więc wartości oddzielone przecinkiem. To format tekstowy do przechowywania tabel, każdy wiersz w pliku to wiersz w tabeli, wartości odzielone przecinkami to wartości kolejnych kolumn. Większość arkuszy kalkulacyjnych (np. MS Excell) potrafi wczytywać tego rodzaju pliki jako tabele. Możesz użyć arkusza, żeby narysować wykres.
Krzywa logistyczna z tego zadania pojawiała się często w mediach na początku pandemii. Spróbuj pobawić się z różnymi parametrami i zasymulować różne scenariusze rozwoju epidemii.

Zadanie 1 – algorytm Neville'a (1 pkt)

Napisz funkcję

```
double neville(const std::vector<double> & px,
        const std::vector<double> & py, double x)
```

która dla danej wartości x znajduje za pomocą algorytmu Neville'a wartość wielomianu interpolacyjnego stopnia px.size()-1 przechodzącego przez px.size() punktów o współrzędnych przekazanych w wektorach px i py. Funkcja może utworzyć (dynamiczną) kopię wektora ya - nie twórz jednak żadnych innych pomocniczych tablic ani kolekcji!
Na przykład następujący program

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

double neville(const std::vector<double> & px,
        const std::vector<double> & py, double x)
{
        //Tutaj napisz swoje rozwiazanie
}

int main()
```

```
{
        //3x^3-4x^2+x-2
        std::vector<double> px ({-1, 0, 2, 3});
        std::vector<double> py ({-10, -2, 8, 46});
        std::vector<double> x{0, -0.5, 1, 2.2};
        for (auto i : x)
        {
                std::cout << "p(" << i << ") = "
                << neville(px, py, i) << std::endl;
        }
}
```

powinien wydrukować nastepujący output:
p(0) = -2
p(-0.5) = -3.875
p(1) = -2
p(2.2) = 12.784

Uwaga: Kompiluj w zgodzie ze standarded C++11 lub nowszym.
Podpowiedź: Funkcja neville() nie musi być rekurencyjna.

Zadanie 2 – współczynniki wielomianu interpolacyjnego (1,67 pkt)

Napisz funkcję

```
void polyCoeff(const std::vector<double> & px,
        const std::vector<double> & py, std::vector<double> & c);
```

która oblicza i umieszcza w wektorze c współczynniki wielomianu interpolacyjnego stopnia px.size()-1 przechodzącego przez px.size() punktów o współrzędnych przekazanych w wektorach px i py. Skorzystaj z funkcji implementującej algorytm Neville'a.
Następnie napisz funkcję

```
void printPoly( std::vector<double> & c);
```

która pobierać będzie wektor współczynników wielomianu (uzupełniany przez funkcję polyCoeff) i wypisywać wielomian w postaci:

$$y = c_0 + c_1 x + c_2 x^2 + ... \tag{3}$$

Przykład:

```
#include <iostream>
#include <vector>
#include <cmath> //abs

double neville(const std::vector<double> & px,
        const std::vector<double> & py, double x)
{
```

```
        // Funkcja z poprzedniego zadania.
}


void polyCoeff(const std::vector<double> &px,
  const std::vector<double> &py, std::vector<double> &c)
{
        // ta funkcja liczy wspolczynniki i wpisuje do c.
}

void printPoly(const std::vector<double> &c)
{
        // ta funkcja ma wypisac wynik w ladny sposob
}

int main() {
        std::vector<double> px{-1, 0, 2, 3};
        std::vector<double> py{1, 1, 7, 25};
        std::vector<double> coef;
        polyCoeff(px,py,coef);
        printPoly(coef);
}
```

Output: **y = 1 - x^1 + x^3**

Podpowiedź: Do usunięcia elementu stojącego na i-tym miejscu (numerowane od zera) w wektorze v, można użyć instrukcji: **v.erase(v.begin()+i);**.