1. **Formulate the statement of the interpolation problem with Cubic Spline [mathematical formula]**

   Given cell space $\Omega_{[a;b]}$ and R_s - mapping from f to $\Omega_{[a;b]}$ - produce an interpolating function I, such that $I(R_s(f)) = R_s(f)$ on $\Omega_{[a;b]}$ and $I(R_s(f)) \in C^2[a;b]$

2. **Formulate the functional and differential compatibility conditions [mathematical formula]**

   $I = f$ on $\Omega_{[a;b]}$ and $I^{(2)} = f^{(2)}$ on $\Omega_{[a;b]}$

3. **Formulate stitching conditions [mathematical formula]**

   For every pair of adjacent sub-splines S_1, S_2 and point of adjacency $x \in \Omega_{[a;b]}$, $S_1^{(1)}(x) = S_2^{(1)}(x)$

4. **Justify why these conditions provide you with the required smoothness [thesis text, no more than 500 characters]**

   Each sub-spline $S_n \in \Omega_{(i,i+1)}$ is produced to be cubic polynomial, thus $I \in C^2[a,b]$, except maybe the junction points $x_i \in \Omega_{[a,b]}$, so let's continuously prove, that in those points $I \in C^0, C^1, C^2$

   $I \in C^0$ at Ω_i because of functional compatibility condition - its value is well defined and is equal to f at that point

   $I \in C^1$ at Ω_i because of stitching condition - the condition equalizes adjacent sub-splines first derivatives at the point, thus implying their existence and definiteness

   $I \in C^2$ at Ω_i because of differential compatibility condition - its value is well defined and is equal to $f^{(2)}$ at that point

   Since there is no point $x_i \in [a;b]$ such that $x_i \notin C^2$, we can state that $I \in C^2[a;b]$

5. **Derive dependency formula: the dependence of the second derivatives at the grid nodes on the increment of the function (the function values difference on the grid nodes). [Mathematical formulas derivation. Detailed, with clear transitions]**

   For a sub-spline $S_i \in [x_i; x_{i+1}]$ with a cubic polynomial formula

   (1) $S_i(x) = a_{0,i} + a_{1,i}(x - x_i) + a_{2,i}(x - x_i)^2 + a_{3,i}(x - x_i)^3$;

   and

   (2) $S_i^{(2)}(x) = 2a_{2,i} + 6a_{3,i}(x - x_i)$

   we have 4 corresponding criteria equations:

   1. $S_i(x_i) = f(x_i)$
   2. $S_i(x_{i+1}) = f(x_{i+1})$
   3. $S_i^{(2)}(x_i) = f^{(2)}(x_i)$

3. $S_i^{(2)}(x_{i+1}) = f^{(2)}(x_{i+1})$

By substituting (1) and (2) in the system and writing it in a matrix format, we will get

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & h & h^2 & h^3 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 2 & 6h \end{bmatrix} \begin{bmatrix} a_{0,i} \\ a_{1,i} \\ a_{2,i} \\ a_{3,i} \end{bmatrix} = \begin{bmatrix} f(x_i) \\ f(x_{i+1}) \\ f^{(2)}(x_i) \\ f^{(2)}(x_{i+1}) \end{bmatrix},$$

where $h = x_{i+1} - x_i$ is a cell step, constant over cell space. Solving this system gives us:

$a_{0,i} = f(x_i)$

$a_{1,i} = \frac{\Delta f_i}{h} - \frac{m_i}{2}h - \frac{\Delta m_i}{6}h$

$a_{2,i} = \frac{f^{(2)}(x_i)}{2}$

$a_{3,i} = \frac{\Delta m_i}{6h}$,

where $m_i = f^{(2)}(x_i)$, $\Delta m_i = m_{i+1} - m_i$, $\Delta f_i = f_{i+1} - f_i$

According to stitching criteria, $S_i^{(1)}(x_{i+1}) = S_{i+1}^{(1)}(x_{i+1})$, or, substituting:

$a_{1,i} + 2a_{2,i}(x_{i+1} - x_i) + 3a_{3,i}(x_{i+1} - x_i)^2 = a_{1,i+1}$

simplified to

$a_{1,i} + 2a_{2,i}h + 3a_{3,i}h^2 = a_{1,i+1}$

expanded to

$\frac{\Delta f_i}{h} - \frac{m_i}{2}h - \frac{\Delta m_i}{6}h + m_i h + \frac{\Delta m_i}{2}h = \frac{\Delta f_{i+1}}{h} - \frac{m_{i+1}}{2}h - \frac{\Delta m_{i+1}}{6}h$

After simplification we are getting an answer to the question, which is

$\frac{h}{6}m_i + \frac{2h}{3}m_{i+1} + \frac{h}{6}m_{i+2} = \frac{\Delta f_{i+1}}{h} - \frac{\Delta f_i}{h}$, $i \in [0, n-2]$

6. **Create a system of equations using this formula [Matrix representation. Mathematical formulas]**

$$\begin{bmatrix} \frac{h}{6} & \frac{2h}{3} & \frac{h}{6} & 0 & \dots & 0 \\ 0 & \frac{h}{6} & \frac{2h}{3} & \frac{h}{6} & 0 & \dots \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & \frac{h}{6} & \frac{2h}{3} & \frac{h}{6} \end{bmatrix} \begin{bmatrix} m_0 \\ m_1 \\ \vdots \\ m_n \end{bmatrix} = \begin{bmatrix} \frac{\Delta f_1}{h} - \frac{\Delta f_0}{h} \\ \frac{\Delta f_2}{h} - \frac{\Delta f_1}{h} \\ \vdots \\ \frac{\Delta f_{n-1}}{h} - \frac{\Delta f_{n-2}}{h} \end{bmatrix}$$

7. **Explain what is an unknown variable in this system. whether the system is closed with respect to an unknown variable. What is missing for closure. [Text, no more than 200 characters]**

$m_i, i \in [0, n]$ are values of $f^{(2)}(x)$ at the adjacency points $x_i \in [0, n]$. The system is open with respect to them, because in the matrix above has $n + 1$ variables(m_i) with only n-1 of equations. We need to define border values m_0 and m_n for the closure.

8. **Bring this matrix to the appropriate form to use the Tridiagonal matrix algorithm [Mathematical derivation. Use Gauss Elimination]**

$$\begin{bmatrix} \frac{2h}{3} & \frac{h}{6} & 0 & 0 & \dots & 0 \\ \frac{h}{6} & \frac{2h}{3} & \frac{h}{6} & 0 & \dots & 0 \\ 0 & \frac{h}{6} & \frac{2h}{3} & \frac{h}{6} & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & \frac{h}{6} & \frac{2h}{3} \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ \vdots \\ m_{n-1} \end{bmatrix} = \begin{bmatrix} \frac{\Delta f_1}{h} - \frac{\Delta f_0}{h} \\ \frac{\Delta f_2}{h} - \frac{\Delta f_1}{h} \\ \vdots \\ \frac{\Delta f_{n-1}}{h} - \frac{\Delta f_{n-2}}{h} \end{bmatrix},$$

or, to remove the right part,

$$\begin{bmatrix} \frac{2h}{3} & \frac{h}{6} & 0 & 0 & \dots & 0 & \frac{\Delta f_0}{h} - \frac{\Delta f_1}{h} \\ \frac{h}{6} & \frac{2h}{3} & \frac{h}{6} & 0 & \dots & 0 & \frac{\Delta f_1}{h} - \frac{\Delta f_2}{h} \\ 0 & \frac{h}{6} & \frac{2h}{3} & \frac{h}{6} & \dots & 0 & \vdots \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 0 & \frac{h}{6} & \frac{2h}{3} & \frac{\Delta f_{n-2}}{h} - \frac{\Delta f_{n-1}}{h} \end{bmatrix}$$

9. **Derive formulas of direct pass and reverse pass of Tridiagonal matrix algorithm [Mathematical formals]**

In the resulting matrix each equation has a form of:

$a_i x_{i-1} + b_i x_i + c_i x_{i+1} - y_i = 0,$

with $y_n = a_1 = 0$, $i \in [1; n]$

After Gaussian elimination, our matrix will be transformed into

$$\begin{bmatrix} 1 & -P_1 & 0 & \dots & Q_1 \\ 0 & 1 & -P_2 & \dots & Q_2 \\ \vdots & \dots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 1 & Q_n \end{bmatrix}$$

with each $x_i = P_i x_{i+1} + Q_i$, and, of course, $x_{i-1} = P_{i-1} x_i + Q_{i-1}$

Substituting it back into the first equation will result in

$a_i(P_{i-1} x_i + Q_{i-1}) + b_i x_i + c_i x_{i+1} - y_i = 0$

If we then leave x_i on the left side and put everything else on the right, we will have

$x_i = \frac{y_i - c_i x_{i+1} - a_i Q_{i-1}}{a_i P_{i-1} + b_i} = \frac{-c_i}{a_i P_{i-1} + b_i} x_{i+1} + \frac{y_i - a_i Q_{i-1}}{a_i P_{i-1} + b_i}$

which fits our previous form $x_i = P_i x_{i+1} + Q_i$ with

$P_i = \frac{-c_i}{b_i + a_i P_{i-1}}$

$Q_i = \frac{y_i - a_i Q_{i-1}}{b_i + a_i P_{i-1}}$

For $i = 0$, $P_i = \frac{-c_i}{b_i + a_i * 0} = \frac{-c_i}{b_i}$, $Q_i = \frac{y_i - a_i * 0}{b_i + a_i * 0} = \frac{y_i}{b_i}$

At point $n, x_n = Q_n = -y_n$, it is obvious from the matrix above

10. **Implement code prototype of the future algorithm implementation. Classes/methods (if you use OOP), functions. The final implementation (on language chosen by you) should not differ from the functions declared in the prototype. [Python code in ipynb]**

    $readFile(filename : String)$

    $getValuesAt(coeffs, fs, xs : Array[double])$

    $getCoeffs(fs, ms : Array[double], h : double)$

    $getMs(fdeltas : Array[double], h : double)$

    $writeFile(filename : String, data : Array[double])$

11. **Derive formula of Cubic Spline method error [Mathematical formulas]**

    $||f^{(p)} - S_3^{(p)}|| = max_{[a;b]}|f^{(p)} - S_3^{(p)}| \leq max_{[a;b]}|f^{(4)}| * h^{(4-p)}$

    No explicit deriving, too weak; Watch 1st lab slides

    That error estimation is based on the assumption, that this $f^{(4)}$ exists, meaning $f \in C^t[a;b], t >= 4$

12. **Rate the complexity of the algorithm [Text, and rate in terms of big O, no more than 100 characters]**

    Direct and forward pass = $O(N)$, everything else - O(1), thus total = $O(N)$