(a)
initial hidden state : **h0 = 0**
initial cell state: **C0 = 0**

Step	Forget Gate	Input Gate	Cell State	Output Gate
1	f1 = σ(Wf * [h0, x1] + bf)	i1 = σ(Wi * [h0, x1] + bi)	C1 = f1 * C0 + i1 * tanh(Wc * [h0, x1] + bc)	o1 = σ(Wo * [h0, x1] + bo)
2	f2 = σ(Wf * [h1, x2] + bf)	i2 = σ(Wi * [h1, x2] + bi)	C2 = f2 * C1 + i2 * tanh(Wc * [h1, x2] + bc)	o2 = σ(Wo * [h1, x2] + bo)
3	f3 = σ(Wf * [h2, x3] + bf)	i3 = σ(Wi * [h2, x3] + bi)	C3 = f3 * C2 + i3 * tanh(Wc * [h2, x3] + bc)	o3 = σ(Wo * [h2, x3] + bo)
4	f4 = σ(Wf * [h3, x4] + bf)	i4 = σ(Wi * [h3, x4] + bi)	C4 = f4 * C3 + i4 * tanh(Wc * [h3, x4] + bc)	o4 = σ(Wo * [h3, x4] + bo)
5	f5 = σ(Wf * [h4, x5] + bf)	i5 = σ(Wi * [h4, x5] + bi)	C5 = f5 * C4 + i5 * tanh(Wc * [h4, x5] + bc)	o5 = σ(Wo * [h4, x5] + bo)

(b)
LSTMs address the vanishing gradient problem by:

- Introducing a cell state that can store information over long sequences.
- Utilizing gates to selectively control the flow of information.
- Employing additive operations in the cell state update, which helps to preserve the gradient signal.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

When calculating the gradient of C_t with respect to C_{t-1} (which is essential for backpropagation), the gradient of C_{t-1} will be directly added to the overall gradient, rather than being multiplied. This additive nature helps to preserve the gradient signal and prevent it from vanishing, even over long sequences.
(c)

- The dimension of each weight matrix: Wx = 100x100, Wh = 64x64
- The total number of parameters : 64 biases for each of 4 gates: 64x4=**256**, and 2 matrices: 100*64+64*64=**6400**+**4096**. Total: **10752**
- The memory requirements during training: assuming number is a float type = 4byte.
  Then: memory for parameters: 10752*4 = **43008B**
  For each sample in a batch + gradients: hidden state + cell state + input: 64B+64B+100B+64B+64B+100B=396B
  For entire batch: 396B*32samples =**12672B**
  Total memory: **43008B + 12672B=55680B = 54.37K**