Biomex Embedded Software Engineer: C++ Programming Exercise

Our machine learning models take motion sensor data as input, with linked lists used to store the data to be passed to the models; each time a new data value becomes available it is stored in a new node that is appended to the list. There is a complication, in that models may or may not be quantized, in which case they accept 8-bit integer-typed data values, as opposed to the 32-bit floating point values accepted as standard. The decision as to whether quantized models will be used is not known until runtime, when a linked list of the required type (*int* in the case of quantization, *float* otherwise) must be constructed dynamically. Once the decision is made, all subsequent nodes of the linked list will hold data of the same type – our linked lists never mix integer and float data. Upon availability of new data, linked lists perform some preliminary processing prior to storage, but the nature of this preprocessing differs between integer- and float- linked lists. For float-typed linked lists, incoming values are simply normalised by subtracting a scalar float32 mean and dividing by a scalar float32 standard deviation. For integer-typed linked lists, incoming values are normalised using the same constants and then converted to 8-bit integers according to the 8-bit quantization specification described here: https://huggingface.co/docs/optimum/concept_guides/quantization. All constants – normalisation mean and standard deviation, and quantization scale and zero-point - are known at compile time.

Write a suitable linked list implementation in C++. You may use the C++ Standard Library, but no external libraries. Do not use templates. The node constructor should accept a 32-bit float scalar data value to which preprocessing is subsequently applied, as described above, prior to storage. Use arbitrary values for the required preprocessing constants (any values you like).

Ideally, your implementation will be written with the following in mind:

- **Minimal code replication**. In future, a lot of methods will be added that are common to both float- linked lists and integer- linked lists (there is no need to implement this common code)
- **Ease of extensibility**. In future, string-typed linked lists may be required too

Write a basic accompanying program that, using your implementation, constructs an integer-typed linked list and appends five nodes. Use arbitrary data values for each node.