SystemC & Behavior Coding

Assignment 4, 2025-10-23

Abstract

Based on the `Node` and `List` classes in Assignment 3, develop a `Node` template base class then derive a `List` template class. Then a `main()` that uses `List`.

Please read carefully and follow all the instructions. All inputs and outputs required are described in the text. And do follow strictly the coding style described in the first class. Five (5) points will be taken for each bug, missing required output behavior, and incorrect coding style.

Notice that some variable names are intentionally changed. You must complete the code and add things like `#ifndef/#endif` in the header file and where needed.

Array, the template base class

Description

1. Create a template class called `Node` with a template data type T, which has one data member:
   - `T *_Node;`
2. And `Node` has access methods (member functions) as follows:
   - `Node();`
     - Initializes `_Node` as NULL
   - `Node(unsigned int _length);`
     - Constructs `_Node` as a `T` array of size `_length`
   - `~Node();`
     - Delete `Node`
   - `T* reCreate(unsigned int _length);`
     - Allocates for `_Node` an integer array of size `_length`
     - Returns the address of newly allocated array

List, the template class derived from Node

Description

1. Create a template class called `List` that is derived from `Node` with the same template class `T`.
2. List has one data member:

- `unsigned int length;`

3. And List has access methods (member functions):
   - `List();`
     - It must calls `Node()` to initialize `_Node`
     - Initializes `length` as 0
   - `List(unsigned int _length);`
     - It must calls `Node(_length)` to initialize `_Node`
     - Initializes `length` as `_length`
   - `List(const List &other);`
     - The copy constructor that copies `other` to `*this`. Notice the base class Array pointer should not be copied. You need to allocate a new `_Node` if `other._Node` is not NULL, i.e. `other` is not an empty list.
   - `~List();`
     - Implicitly calls `~Node()`
     - Resets `_length` to 0
   - `int setLength(unsigned int);`
     - If the original length is 0, the function sets a new `length`, uses `reCreate()` to allocates an array of size `length` to `_Node`, then returns 1.
     - If the original `length` is not 0, the function prints an error message then returns 0.
   - `unsigned int getLength();`
     - The function returns the value of length.
   - `int setElement(unsigned int pos, T val);`
     - Assigns `val` to `_Node[pos]`.
     - The function returns 1 if `pos` is legal; otherwise prints an error message and returns 0.
   - `T getElement(unsigned int pos);`
     - Returns the value of `_Node[pos]` if `pos` is legal.
     - If `pos` is illegal, use an assert to check for this condition, prints an error message then exit the program.

For below access functions please refer to the lecture notes Section 3-1

- `List& operator=(const List &);`
- `inline bool operator==(const List &);`
- `friend ostream& operator<<(ostream &, List);`

int main(int argv, char *argv[])

Description

1. Create two input files, one of data type `double` and the other type `string`. The first line of each file is the length of the data. For example:
   3
   2.2348
   53.2
   0.2772
2. Since it is difficult to check if the input is of `double` type, and any data can be considered `string` type and thus no use in checking `string`'s legitimacy. In this assignment you can assume all input data are legal and develop no checking mechanism.
3. When executing your program, the first input argument, `argv[1]`, is the `double` file name. And `argv[2]` is the `string` file name. There should have no other input arguments (use `argc` to verify.)
4. Instantiate one `double List`s and two `string List`s. Use data in the `double` file to construct the `double List`. And use the data in the `string` file to construct the first `string List`.
5. Use copy constructor to copy the already constructed `List` to create a second `double List`.
6. Use assignment operator to assign the first `string List` to the second.
7. Use `operator==` to check if the two `double List` and `string List` are indeed the same. Use `cout` to report the comparison result.
8. Use `setElement()` to assign values to all `List` objects.
9. Modify the `makefile` from the Assignment 3 to compile your program.

Please turn in the source codes, `Node.h`, `List.h`, `main.cpp` and `makefile` only. Make necessary changes to the `makefile` provided in Assignment 3. Do not turn in the executable. Make use of code generator AI as much as possible so you can complete the assignment in time. Though you can choose to code all by yourself, it is also crucial to learn how to make use of AI to help you work efficiently. Thoroughly verify the AI-generated code using the verification skills we discussed in the first class.

Due date

3:00 PM, November 6, 2025

Score weight (towards the final grade) 5%