CSE211 DATA STRUCTURES

LAB 1 FALL 2024

LINKED LIST

Prerequisites

Open the Ubuntu terminal and type the following after downloading the `tarball` file:

```
cd /mnt/c/Users/user/Downloads && tar -xvf lab1_3.tar.gz --one-top-level=lab1_3
&& cd /mnt/c/Users/user/Downloads/lab1_3 && make all
code .
```

Introduction

In this lab, you will implement various operations on a singly linked list using C++. The linked list is implemented as a template class `LinkedList` that stores elements of type `T`. Your task is to complete the implementation of the following functions in the `LinkedList` class:

1. `reverseInGroups`: Reverse the linked list in groups of K nodes.
2. `swapPairs`: Swap adjacent pairs of nodes in the linked list.
3. `mergeSorted`: Merge two sorted linked lists into a single sorted list.
4. `partitionList`: Partition the list around a value x, such that all nodes less than x come before nodes greater than or equal to x.

Project Structure

The project has the following structure:

```
.
├── bin // executable files will be generated here by the compiler
|   └── linkedlist
├── include // header files
|   └── LinkedList.hpp
├── obj // object files will be generated here by the compiler
|   └── LinkedList.o
|   └── main.o
├── src // source files
|   └── LinkedList.cpp
|   └── main.cpp
├── instructions.pdf // instructions for the lab
├── Makefile // Makefile to build and run the project
```

- `include/LinkedList.hpp`: Header file containing the declaration of the `LinkedList` class and its member functions.
- `src/LinkedList.cpp`: Source file where you will implement the member functions of the `LinkedList` class.
- `src/main.cpp`: Main source file that demonstrates the usage of the `LinkedList` class.

- `Makefile`: Makefile to build and run the project.

Instructions

1. Navigate to the `src/LinkedList.cpp` file.
2. Implement the following functions in the `LinkedList` class:
    - `reverseInGroups(std::size_t k)`: Reverse nodes in groups of k. If the last group has fewer than k nodes, reverse them too.
    - `swapPairs()`: Swap adjacent pairs of nodes in the list.
    - `mergeSorted(LinkedList<T>& other)`: Merge two sorted lists while maintaining order.
    - `partitionList(const T& x)`: Partition list around value x, maintaining relative order in each partition.
3. Refer to the function documentation comments in the `include/LinkedList.hpp` file for more details on each function.
4. You can use the `main.cpp` file to test your implementation manually.
5. To compile and run the project, use the provided `Makefile`:

```
make all
make run
```

Hints

- For `reverseInGroups`, process the list in groups of k nodes, reversing each group separately.
  Example: For k=3, [1,2,3,4,5,6,7,8] becomes [3,2,1,6,5,4,8,7]
- For `swapPairs`, swap adjacent nodes while maintaining proper links.
  Example: [1,2,3,4,5,6] becomes [2,1,4,3,6,5]
- For `mergeSorted`, compare nodes from both lists and link them in sorted order.
  Example: Merging [1,3,5] and [2,4,6] becomes [1,2,3,4,5,6]
- For `partitionList`, maintain two separate lists for values less than x and greater/equal to x.
  Example: For x=5, [3,7,8,5,2,1] becomes [3,2,1,7,8,5]

Evaluation

Your implementation will be evaluated based on:

- Correctness: Functions produce expected output
- Efficiency: Optimal time complexity
- Code quality: Well-organized and readable code

Warning

- Do not modify any files except `LinkedList.cpp`
- Implement only the specified functions
- Do not add additional files or directories
- Do not use global variables
- Do not use additional libraries

- Any **cheating** will result in a **0** score

Good luck with the lab!