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_4.tar.gz --one-top-level=lab1_4
cd /mnt/c/Users/user/Downloads/lab1_4 && 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. `reverseAlternateK`: Reverse alternate groups of K nodes in the linked list.
2. `segregateEvenOdd`: Segregate even and odd numbers in the linked list.
3. `foldList`: Fold the linked list from the middle.
4. `sortByFrequency`: Sort the list by frequency of elements.

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:
   - `reverseAlternateK(std::size_t k)`: Reverse alternate groups of k nodes. If a group has fewer than k nodes, leave it as is.
   - `segregateEvenOdd()`: Arrange nodes so that all even numbers appear before odd numbers while maintaining relative order.
   - `foldList()`: Fold the list from the middle (like folding a paper).
   - `sortByFrequency()`: Sort elements based on their frequency of occurrence (higher frequency first).
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 `reverseAlternateK`, reverse groups of k nodes alternately, leaving other groups unchanged.
  Example: For k=3, [1,2,3,4,5,6,7,8,9] becomes [3,2,1,4,5,6,9,8,7]
- For `segregateEvenOdd`, maintain two separate lists for even and odd numbers.
  Example: [1,2,3,4,5,6] becomes [2,4,6,1,3,5]
- For `foldList`, find middle point, reverse second half, and merge alternately.
  Example: [1,2,3,4,5] becomes [1,5,2,4,3]
- For `sortByFrequency`, count frequencies and sort based on count.
  Example: [2,3,2,4,5,2,4,3] becomes [2,2,2,3,3,4,4,5]

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!