Assignment 2

Linked Lists

Submission Notes

This assignment requires one cpp file submission. Do not upload a code snippet, the only files you submit for the code should be the source files. Do not submit any zip or compressed files, binary files, or any other generated files. Do not put the code inside the PDF file. Submission of unnecessary files or binary files in addition to source files will make you lose the points of the assignment. The code must have the following:
a. Use consistent indentation through your code.
b. Use consistent braces style through your code.
c. File-level comments, these comments will include the file name, your name, course number, and date of last modification.
d. Function level comments include the function's description, the function's parameters, and the function's return value. These comments will appear before each of the functions, not the prototypes.
e. In function comments, these comments will include a description of the atomic functionalities within the bodies of the functions.
f. We evaluate your code using the following C++ online compiler. You will not get points if your assignment does not compile with this.

`https://www.onlinegdb.com/online_c++_compiler`

SinglyLinked List functions (100 points)

1) Node* removeAll(Node* head, int val) 40 points

Implement a removeAll(Node* head, int val) function that removes all elements with a given value from a linked list with the given head. The function should return the new head of the linked list. Here is the hint:

- Write a while loop and check the head nodes till a head's data$\neq$ val.
- Create a prevNode pointer and set it to be the new head of the list.
- Write a while loop and check the non-head nodes. Make sure to check the non- head nodes only if the prevNode is not NULL.

Example:
Linked list is: 10− > 15− >10− > 20− > NULL
head = removeAll(head, 10);
Linked list after insertToPlace is: 15− > 20− > NULL

2) Node* insertToPlace(Node* head, int val, int place) 40 points

Implement a function that takes in a pointer to the head of a linked list, a value to insert into the list, val, and a location in the list in which to insert it, place, which is guaranteed to be greater than 1, and does the insertion. If place number of items are not in the list, just insert the item in the back of the list. You can assume that the linked list into which the inserted item is being added is not empty. The function should return the head of the list. Here is the hint:

- If the linked list is empty or place <= 1 no action and return the head
- Before insertion create a temp node and fill-up the data
- Use a counter variable and traverse the list until you reach to the end or you reach to the place
- As soon as you stop traversing, add the temp node after that.

Example:
Linked list is: 30− > 15− > 20− > NULL
head = insertToPlace(head, 44, 2);
Linked list after insertToPlace is: 30− > 44− > 15− > 20− > NULL

3) Test your program 20 points

Implement a test program using the functions we have seen in the class to see if your removeAll and insertToPlace functions work fine. Your program should have the following components implementations:

1. class Node
2. void insertBeg(Node** headRef, int newData)
3. void printList(Node *node)
4. Node* removeAll(Node* head, int value)
5. Node* insertToPlace(Node* head, int val, int place)
6. int main()

You can copy components 1 2, and 3 from LLfunctions.cpp code in canvas. You need to implement component 4 and 5 by your own. For component 6, you need to modify the main function that we have seen in the class to test your program.