National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

Assignment 1: Implement Various Operations of Singly Linked Lists

Data Structures & Algorithms

Section: BSCS-10-A&BC **Session:** Fall-2022
Instructor: Dr. Yasir Faheem **Mapped to**: CLO-1

Sally, a software engineer, has been hired to work part-time to develop a dictionary application for the Oxford English Dictionary. Your task is to help Sally develop a basic version of the dictionary by using a singly linked list.

You will currently work on a smaller dataset. The dataset is provided to you in .txt file along with the assignment files. It's a simple text file that contains words followed by their meanings. An example is shown below:

```
introspection the observation of one's own mental and emotional processes
philanthropist  a person who seeks to promote the welfare of others
antidote  a medicine taken or given to counteract a particular poison
.           ...
.           ...
.           ...
```

Likewise, you have also been provided with a parser that will read the text file:
Note: You may read the xxx tutorial to learn file handling in C/C++.

```
void readFileData() //reads text from a file. You should add functionality
to store the fetched data into a singly linked list.
{
  Dictionary *dict;
  string word, meaning;
  fstream fin;
  cout << "\e[46mEnter the filename\x1b[0m ";
  cin.ignore();
  cin >> filename;
  fin.open(filename);
  cout << "\e[0;32mFile reading succesful.\x1b[0m\n";
  while( fin >> word )  //write into file name
  {
    fin.ignore();
    getline(fin,meaning);
    dict = new Dictionary(word,meaning);
    cout << *dict << endl; //output newly created Dictionary object
    //insert the new node to the linked list here
  }
  fin.close();
}
```

And another function that will save your linked list as a text file:

```
void writeToFile(ListNode *headNode) //write the linked list to a text
file
{
  string word, meaning;

  fstream dictFile; //create fstream object for the file
  cout << "\e[46mEnter the filename\x1b[0m ";
  cin.ignore();
  cin >> filename;
  dictFile.open(filename, std::ios::app); //create/open a text file in
append mode. new information is always added to the end

  ListNode *iterator = headNode;

  while(iterator != NULL) {
    word = ListNode->data.word;
    meaning = ListNode->data.meaning;
    dictFile << word;
    dictFile << " " << meaning << endl; //write to data file
    iterator = iterator->next;   //advance to next node
  }
  dictFile.close();
}
```

1. Modify the above readFileData() function inside the Parser.h file to insert the parsed dictionary objects into a singly linked list.

2. Similarly, modify the writeToFile(ListNode *headNode) function inside the Parser.h file to convert a linked list into a text file.

3. Implement a function that prints all nodes of a **sinlgy linked list** in the **reverse order** according to the dictionary words. Note that you are not allowed to copy the contents of the list in any other structure. The stream output operator has already been overloaded for you in the provided Dictonary class. You simply have to pass the dictionary object(s) into an output stream for printing.

```
void printReverse(LinkedList list) {
  //print the list in reverse order
}
```

4. Implement a function which **reverses** the order of **a linked list** according to the **dictionary words**. Note that this function should only change the pointer field in all nodes of a list; you are not allowed to modify data field of any node.

```
ListNode *reverseList(ListNode *start, ListNode *end) {
  //returns a linked list that is the reverse of the original list
}
```

5. Implement a function that filters all those dictionary nodes from the linked list which have the inputted letter in their data part and returns a new list. For instance, if the original list is hardwork-> sunlight->planetary->equilibrium, and the inputted letter is 'u', then, the updated list should be sunlight->equilibrium. Your function should **delete** the unmatched words i.e., change the pointer fields and then delete the pointer itself.

   Note: Include all checks for various input cases like: a) all elements are a match, b) only the last element is a match, c) the list has only node which is a match, d) both first and last nodes match

```
ListNode *filterByLetters(ListNode *start, string letter) {
  //returns a linked list that contains only those words that have the
specified letter
}
```

6. Implement a function that **rearranges** a linked list such that all the even length words appear at the start of the list, and all the odd length words appear at the end of the list. For example, for the list one ->three->four->five->six->seven. The resulting list should be four->five->one->three->six->seven.

```
ListNode *rearrangeByEvenOddLength(ListNode *start) {
  //rearrange the list so that all the words with even length are at
the front
  //and all the words with odd length are at the back
  //return the list
}
```

Deliverables:

- Ensure that you use the provided function definitions in your source code. **DO NOT** rename the functions or their input parameters.
- You need to use the the writeToFile(ListNode *headNode) function to create output files for each of your function. If an output file isn't applicable in some case, upload the screenshot(s) in a word document.
- Moreover, you should upload your source code on GitHub as instructed. Ensure that you perform frequent commits to the repository each time you work on the assignment.

Extra Credit:

- Read the xxxx tutorial on generic data types.
- Create a template linked list class and implement all the functionalities using that class. You do not have to submit duplicate functions. If you are working with a template class, only

submit the code you have written for the template class.

- Use of recursion.