**ECE 250 – Project 1**

**Design Document**

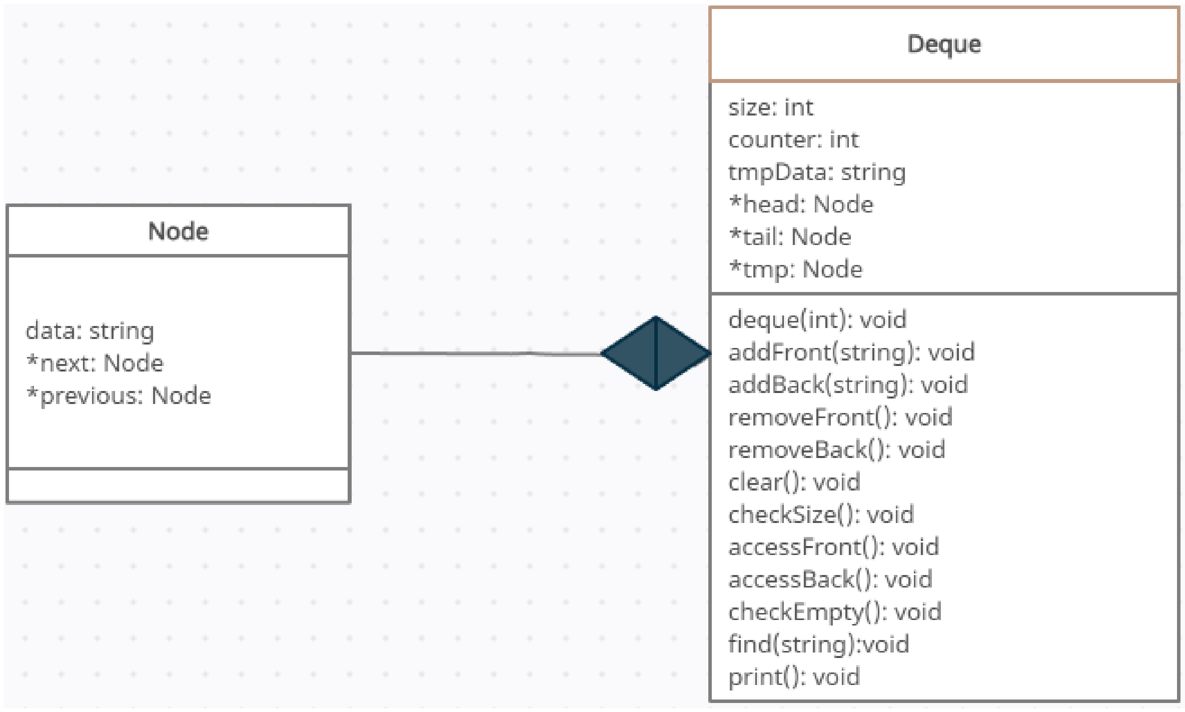
**Calvin Zhao 20899433**

**October 17/2022**

## 1. Overview of Classes

- **Class Node**: Node objects that are used to create a double linked list. The nodes created will hold data of type string and a pointer of the addresses towards the next and previous nodes. Also the use of 'friend class' will allow other class to access the private variables of this node class.
    - Member Variables:
        - data (data type: string) -------------------------stores information
        - next(data type: Node pointer) --------------- address to next node
        - previous(data type: Node pointer)-----------address to previous node
    - Member Functions:
        - Node: a Constructor
        - ~Node: a Destructor
- **Class Deque:** A data structure that is created using a doubly linked list. It is a dynamic data structure that will store the data in a list of connected nodes performing functions such as adding and deleting elements at the front and end of a linked list and many more.
    - Member variables:
        - size(data type: int)-------------------------------max size of the deque
        - counter (datatype: int)--------------------------- current size of list
        - tmpData(datatype: string)---------------------- temporary data
        - tmp (datatype: Node pointer)----------------- temporary pointer
        - head (datatype: Node pointer)---------------- address to first element in the deque
        - tail (datatype: Node pointer)------------------- address to last element in the deque
    - Member functions:
        - addFront, addBack: add element to front or back of list
        - removeFront, removeBack: remove element from front or back of list
        - clear: clear all elements from list
        - checkSize: return size of element
        - accessFront, accessBack: return the element at the front or back of list
        - checkEmpty: check if there is any elements in the list
        - find: searches for a specific URL in the list
        - print: outputs all elements from back of the deque to the front

2. **UML Class Diagram**

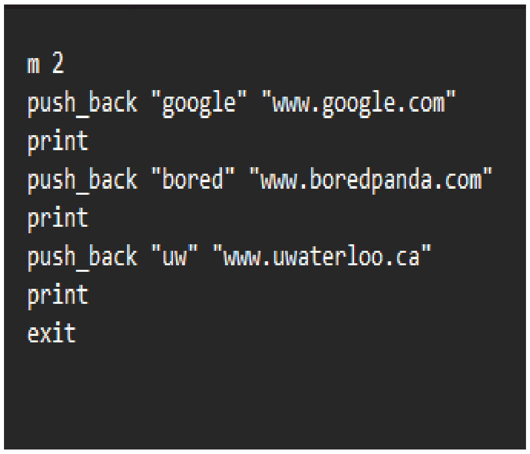


Since deque class is the parent and the node is the child, a diamond shape is used from the Node class to the Deque class.

3. **Details on design decisions**
   - **Class Node:**
     - **Constructor**: initializes the data as an empty string and have the next and previous pointer set as nullptr
     - **Destructor**: Frees the memories that are used by the next and previous pointers by deleting them and setting them to NULL.
   - **Class Deque**:
     - **Constructor**: Initializes the maximum size of the deque and set head and tail to nullptr
     - **Destructor**: Free up the memory of the head and tail by setting them to NULL
4. **Test Cases**

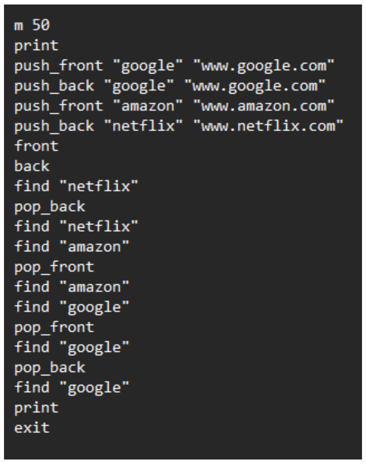
   Used test cases that are provided on the course site: test01.in, test02.in, and test06.in. Test1 covered if my functions that add elements to the front and back functioned correctly if the deque is full. Test2 covered what happens if two of the same url is added and covered if my find and remove front and back functions worked correctly if there wasn't any elements in the deque.

Test1 example:



```
m 2
push_back "google" "www.google.com"
print
push_back "bored" "www.boredpanda.com"
print
push_back "uw" "www.uwaterloo.ca"
print
exit
```

Test2 example:



```
m 50
print
push_front "google" "www.google.com"
push_back "google" "www.google.com"
push_front "amazon" "www.amazon.com"
push_back "netflix" "www.netflix.com"
front
back
find "netflix"
pop_back
find "netflix"
find "amazon"
pop_front
find "amazon"
find "google"
pop_front
find "google"
pop_back
find "google"
print
exit
```

5. **Performance**

   All other member functions except find, clear, print ran at O(1). To achieve O(1), I made it a priority to not use any loops or iterations to perform the tasks for adding and removing elements to the front and back . Since find, clear, and print requires iterating through the deque to complete the purpose of those functions, the runtime for those functions is O(n).