CS214 – Data Structures
Assignment #2

Linear Structures

- Implement an **Array-Based List** with the following methods:

    - insert (elementType element) : void
    - insertAt (elementType element, int index) : void
    - retrieveAt (int index): elementType
    - removeAt (int index): void
    - replaceAt (elementType newElement, int index)
    - isItemAtEqual (elementType element, int index) : bool
    - isEmpty () : bool
    - isFull () : bool
    - listSize () : int
    - maxListSize (): int
    - clear (): void
    - print () : void

- Implement a **Single Linked List** with the following methods:

    - insertAtHead (elementType element) : void
    - insertAtTail (elementType element) : void
    - insertAt (elementType element, int index) : void
    - removeAtHead () : void
    - removeAtTail () : void
    - removeAt (int index) : void
    - retrieveAt (int index): elementType
    - replaceAt (elementType newElement, int index)
    - isExist (elementType element) : bool
    - isItemAtEqual (elementType element, int index) : bool
    - swap (int firstItemIdx, int secondItemIdx) : void // swap two nodes without swapping data.
    - isEmpty () : bool
    - linkedListSize () : int
    - clear (): void
    - print () : void

CS214 – Data Structures
Assignment #2

- Implement a **Doubly Linked List** with the following methods:

    - insertAtHead (elementType element) : void
    - insertAtTail (elementType element) : void
    - insertAt (elementType element, int index) : void
    - insertAfter (* prev_node, int data) : void
    - removeAtHead () : void
    - removeAtTail () : void
    - removeAt (int index) : void
    - retrieveAt (int index): elementType
    - replaceAt (elementType newElement, int index)
    - isExist (elementType element) : bool
    - isItemAtEqual (elementType element, int index) : bool
    - swap (int firstItemIdx, int secondItemIdx) : void // swap two nodes without swapping data.
    - reverse () : void //reverse the data in the double linked list
    - isEmpty () : bool
    - doubleLinkedListSize () : int
    - clear (): void
    - forwardTraversal () : void //Print from head to tail
    - backwardTraversal () : void //Print from tail to head

- Implement a **Circular Linked List** with the following methods:

    - insertAtHead (elementType element) : void
    - insertAtEnd (elementType element) : void
    - insertAt (elementType element, int index) : void
    - removeAtHead () : void
    - removeAtEnd () : void
    - removeAt (int index) : void
    - retrieveAt (int index): elementType
    - replaceAt (elementType newElement, int index)
    - isExist (elementType element) : bool
    - isItemAtEqual (elementType element, int index) : bool
    - swap (int firstItemIdx, int secondItemIdx) : void // swap two nodes without swapping data.
    - isEmpty () : bool
    - circularLinkedListSize () : int
    - clear (): void
    - print () : void

- Implement a **Stack** with the following methods:
    - push (elementType element) : void
    - pop () : elementType element //return the first element and remove it.
    - top () : elementType element //return the first element without removing it.
    - isEmpty () : bool
    - stackSize () : int
    - clear (): void
    - print () : void

- Implement a **Queue** with the following methods:
    - enqueue (elementType element) : void
    - dequeue () : elementType element //return the first element and remove it.
    - first () : elementType element //return the first element without removing it.
    - isEmpty () : bool
    - queueSize () : int
    - clear (): void
    - print () : void

Problems

- By using the previous implemented **Linked List** solve the following problems:

1. Combine Nodes Between Zeros

- You are given a head of a linked list containing a sequence of integers separated by 0s.
- The value presented at the linked list's beginning and end is 0.
- Merge all nodes between any two consecutive 0s into a single node whose value is the total of all the merged nodes.
- There are no two consecutive nodes with value == 0.
- There should be no 0s in the new list.
- Example:
  If the **Input** is the head of the following Linked List: [0, 3, 1, 0, 4, 5, 2, 0],
  Then the Linked List will: [4, 11]

  If the **Input** is the head of the following Linked List: [0, 1, 0, 3, 0, 2, 2, 0],
  Then the Linked List will: [1, 3, 4]

2. Merge K Sorted Linked Lists

- You have **k** linked-lists, each linked-list is sorted in ascending order.
- You need to merge all the given linked-lists into one sorted linked-list then return it.
- Example:
  If the **Input** is an array of pointers storing the head nodes of following the linked lists: [1, 4, 5], [1, 3, 4], [2, 6].
  Then the **Output** Linked List will be: [1, 1, 2, 3, 4, 4, 5, 6]

  If the **Input** is an array of pointers storing the head nodes of following the linked lists: [1, 3, 5, 7], [2, 4, 6, 8], [0, 9, 10].
  Then the **Output** Linked List will be: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10].

- By using the previous implemented **Stack** solve the following problems:

1. Convert the Infix Expression to Postfix Expression

- **Infix expression:** The expression of the form "a operator b" (a + b) i.e., when an operator is in-between every pair of operands.
- **Postfix expression:** The expression of the form "a b operator" (ab+) i.e., When every pair of operands is followed by an operator.
- You have a string of an infix expression as an input and you need to return its postfix expression.
- Example:
  If the **Input** is: X^Y / (5*Z) + 2
  Then the **output** will be: XY^5Z*/2+

  If the Input is: A + B * C + D
  Then the output will be: A B C * + D +

  If the Input is: (A + B) * (C + D)
  Then the output will be: A B + C D + *

2. Longest Valid Parentheses

- You will have a string containing just the characters '(' and ')' as an input.
- You have to return the length of the longest valid (well-formed) parentheses substring (A substring is a contiguous non-empty sequence of characters within a string.).
- Example:
  If the **Input** is: '(()'
  Then the **Output** will be: 2 – as the longest valid parentheses are ().
  If the **Input** is: ''
  Then the **Output** will be: 0.
  If the **Input** is: ')()())'
  Then the **Output** will be: 4 – as the longest valid parentheses are ()().

- By using the previous implemented **Queue** solve the following problems:

1. Generate Binary Numbers from 1 to N

- Write a function that **generates** and **prints** all binary integers from **1** to **N**.
- Example:
  If the **Input** is: 3
  Then the **output** will be: 1 10 11
  If the **Input** is: 5
  Then the **output** will be: 1 10 11 100 101

2. Implement a Stack

- You need to design a stack that supports **push** and **pop** operations using the **enqueue** and **dequeue** operations of the queue.
- You can use one or more queue.

3. Sorting a Queue

- You will have a queue with random integer elements as an input.
- You have to sort it.
- Example:
  If the **Input** is a queue that have: 3 15 2 4
  Then you will **modify it** to be: 2 3 4 15
  If the **Input** is a queue that have: 9 1 2 1
  Then you will **modify it** to be: 1 1 2 9

CS214 – Data Structures
Assignment #2

About the Assignment

1- All the team members must understand all the assignment problems.

2- All the code must be in C++.

3- Any cheating in any part of the assignment is the responsibility of the whole team, and all of the team members will be punished.

4- The solution should compile, run without run-time errors, and handle all the cases.

Submission Rules

1- Assignment is submitted in teams of 3 from any group.

2- You will upload a zipped folder that contains your code (Don't include any .exe files in your submission).

3- Assignment submission is on Google Classroom (No submission through mail).

4- Follow this convention for naming your folder: ID1_ID2_ID3_A#_G# (i.e 20200111_20200222_20200333_A2_G5_G6)

5- Deadline of the Assignment: 20 April, 2023, at 11:59 p.m.

Failure to follow any of the above rules will result in your submission being discarded and your team being considered to have not submitted.