**COMSATS** **University Islamabad, Lahore Campus**

**Department of Computer Science**



**Assignment 1 – Semester SPRING 2023**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Course Title: | Data Structures & Algorithms | | | | | Course Code: | | CSC211 | Credit Hours: | 4(3,1) |
| Course Instructor/s: | Mr. Imran Latif | | | | | Program Name: | | BCS | | |
| Semester: | 4th | Section: | A&B | | Batch | FA21  60 Minutes | | | | |
| Total Marks: | **10** Obtained Marks: | Obtained Marks: | |  | | Due Date: | March 20, 2023 | | | |
| Student’s Name: |  | | | | | Reg. No. |  | | | |
| **Important Instruction:**   * Student is himself/herself responsible for successful submission of assignment on Microsoft Teams. * Your submission must include the following in a single pdf file.   1. Code of all classes  2. Snapshot of the output of submitted code.   * Copied assignment will get zero credit. * Deadline: March 20, 2023 till 11:30 PM | | | | | | | | | | |

***CLO: <3>; Bloom Taxonomy Level: <Applying>***

Q-1 Write and test this method:

Node copy(Node list)

//returns: a new list that is a duplicate of the specified list;

Note that the new list must be completely independent of the specified list. Changing one list should have no effect upon the other.

**public Node copy(Node list){**

**if(list == null)**

**throw new RuntimeException("Linked List is Empty");**

**Node copyHead = new Node(list.data);**

**Node temp = copyHead;**

**while(list.next != null){**

**list = list.next;**

**temp.next = new Node(list.data);**

**temp = temp.next;**

**}**

**return copyHead;}**

Q-2 Write and test this method:

Node concat(Node list1, Node list2)

//returns: a new list that contains a copy of list1, followed by a copy of list2;

For example, if list1 is {22,33,44,55} and list2 is {66,77,88,99}, then concat (list1,list2) will return the new list {22,33,44,55, 66,77,88,99}.

Note that the three lists should be completely independently of each other. Changing one list should have no effect upon the others.

**public Node concat(Node list1, Node list2){**

**Node copyHead = new Node(list1.data);**

**Node temp = copyHead;**

**while(list1.next != null){**

**list1 = list1.next;**

**temp.next = new Node(list1.data);**

**temp = temp.next;**

**}**

**temp.next = new Node(list2.data);**

**temp = temp.next;**

**// System.out.println("Value"+list2.data);**

**while(list2.next != null){**

**list2 = list2.next;**

**temp.next = new Node(list2.data);**

**temp = temp.next;**

**}**

**return copyHead;**

**}**

Q-3 Write and test this method:

void replace (Node list, int I, int x)

//replaces the ith element with x;

For example, if list is {22,33,44,55, 66,77,88,99}, then replace (list, 2, 50) will change list to

{22,33,50,55, 66,77,88,99}.

**public void replace(Node list, int I, int x){**

**Node temp = list;**

**int count = 0;**

**while(temp != null){**

**count++;**

**if(count == I){**

**temp.data = x;**

**}**

**temp = temp.next;**

**}**

**}**

Q-4 Give an algorithm for finding the second-to-last node in a singly linked list in

**public void secondLastNode() {**

**Node current = head, previous = null;**

**while (current.next != null) {**

**previous = current;**

**current = current.next;**

**}**

**System.out.println(previous.data);**

**}**

Q-5 Give an implementation of the size( ) method for the CircularlyLinkedList class,

**public int size(Node list){**

**Node temp = list;**

**int size = 0;**

**if(temp == null)**

**throw new RuntimeException("Linked List is empty");**

**else if(temp.next == list) {**

**size++;**

**return size;**

**}else {**

**do {**

**size++;**

**temp = temp.next;**

**} while (temp != list);**

**}**

**return size;**

**}**

assuming that we did not maintain size as an instance variable.

Q-6 Implement a rotate(int d ) method in the SinglyLinkedList class, which rotates the given linked list at specified index.

Given linked list

10 20 30 40 50 60

Where d=4

Rotated Linked list

50 60 10 20 30 40

**public void rotate(int d){**

**Node current = head, previous = null, newHead = null;**

**int nodeCount = 0;**

**while(current.next != null){**

**if(nodeCount == d){**

**newHead = current;**

**previous.next = null;**

**}**

**previous = current;**

**current = current.next;**

**nodeCount++;**

**}**

**current.next = head;**

**head = newHead;**

**}**