**CENTRE FOR DEVELOPMENT OF ADVANCED COMPUTING**

**Advanced Computing Training School**

Course Name: PG Diploma in Advanced Computing

Batch: September 2021

Module Name: Algo & DS using Java Date: 06-12-2021

Student Name: Md Farazul Haque Max Marks: 40 Marks

PRN No.:210980420078 Duration :2 Hours

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Q1. Using Object oriented concept develop a software in Java to find transpose of matrix. [15 Marks]

Q2. Implement Insertion and deletion of Node at first, and last in a Singly Linked List. [15 Marks]

**Evaluation of Lab Exam should be based on the following criteria:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Criteria** | **Details** | **Max Marks** | **Marks Obtained** |
| Algorithm | Documentation of Algorithm and Flowchart | 30 |  |
| Program adheres to the algorithm and flowchart |
| Efficiency | Program is using only the required number of variables  /conditions/loops/pointers etc and is optimal |
| Correctness | The program produces desired output for a given input |
| The program handles all valid and Invalid inputs |
| Software engineering Principals | The program has meaning variable/function names |
| The program is commented properly (At least 20% of the  code should be commented) |
| Viva |  | 10 |
|  | **Total Marks** | **40** |  |

Md Farazul Haque \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Signature of Student Signature of Evaluator Signature of Coordinator**

**Q1.**

|  |
| --- |
| **package** com.faraz;  **import** java.util.Scanner;  **public** **class** TransposeMatrix {  // initialise variables to store data  **int** r, c;  **int**[][] matrix;  **int**[][] transposedMatrix;  Scanner sc = **new** Scanner(System.***in***);  // get row and column size on object creation  **public** TransposeMatrix() {  System.***out***.println("Enter row size: ");  r = sc.nextInt();  System.***out***.println("Enter column size: ");  c = sc.nextInt();  // create two matrix, one for original matrix and one for transposed matrix  matrix = **new** **int**[r][c];  transposedMatrix = **new** **int**[r][c];  }  // method to insert item in the matrix  **public** **void** insertdata() {  // nested loop to get user input data  **for** (**int** i = 0; i < r; i++) {  **for** (**int** j = 0; j < c; j++) {  System.***out***.print("matrix[" + i + "][" + j + "] = ");  matrix[i][j] = sc.nextInt();  }  }  }  // method to transpose matrix  **public** **void** findTranspose() {  **for** (**int** i = 0; i < r; i++) {  **for** (**int** j = 0; j < c; j++) {  // swap i with j and j with i to get transposed data  transposedMatrix[j][i] = matrix[i][j];  }  }  }  // method to show original matrix  **public** **void** showOriginalMatrix() {  System.***out***.println("\n-----------Original Matrix-----------");  **for** (**int** i = 0; i < r; i++) {  **for** (**int** j = 0; j < c; j++) {  System.***out***.print(matrix[i][j] + " ");  }  System.***out***.println();  }  }  // method to show transposed matrix  **public** **void** showTransposedMatrix() {  System.***out***.println("\n-----------Transposed Matrix-----------");  **for** (**int** i = 0; i < r; i++) {  **for** (**int** j = 0; j < c; j++) {  System.***out***.print(transposedMatrix[i][j] + " ");  }  System.***out***.println();  }  }  } |

Q2.

|  |
| --- |
| **package** com.faraz;  **public** **class** SingleLL {  // create first node/ head node  Node head;  // on object creation set head to null  **public** SingleLL() {  head = **null**;  }  // method to insert data at beginning  **public** **void** insertionAtFirst(**int** data) {  // check list is empty or not  **if** (head == **null**) {  // if empty then create new node with data and store in head  head = **new** Node(data);  } **else** {  // else create new node with data  Node first = **new** Node(data);  // and then link it to first node i.e head of sll  first.link = head;  // and then set head to first  head = first;  }  System.***out***.println(data + " inserted at beginning...");  }  **public** **void** deletionAtFirst() {  // check is sll is empty or not  **if** (head == **null**) {  // if empty then print list empty  System.***out***.println("List is empty...");  } **else** {  // else store the first data into new variable  **int** first = head.data;  // then create temporary node to store head  Node temp = head;  // and then set head to next node i.e head.link  head = head.link;  // and then set null to temporary variables  temp.link = **null**;  temp = **null**;  System.***out***.println(first + " deleted from beginning...");  }  }  **public** **void** insertionAtLast(**int** data) {  // check if list is empty or not  **if** (head == **null**) {  // if empty then create new node and set it to head  head = **new** Node(data);  } **else** {  // else iterate till last node  Node last;  **for** (last = head; last.link != **null**; last = last.link)  ;  // and then set link of last node to new node of data  last.link = **new** Node(data);  }  System.***out***.println(data + " inserted at Last...");  }  **public** **void** deletionAtLast() {  // check if list is empty or not  **if** (head == **null**) {  // if empty then print list empty  System.***out***.println("List is empty...");  } **else** {  // else create two node to iterate till last node and second last node  Node pr = **null**;  Node tr = head;  **for** (tr = head; tr.link != **null**; tr = tr.link) {  pr = tr;  }  // and then set data to last data  **int** lastData = tr.data;  System.***out***.println(lastData + " deleted from Last...");  // and set link of second last node to null  pr.link = **null**;  // and also last node to null  tr = **null**;  }  }  // method to print items stored in list  **public** **void** show() {  System.***out***.print("Items in Single Linked List: ");  **if** (head == **null**) {  System.***out***.println("List is empty");  } **else** {  Node tr;  **for** (tr = head; tr != **null**; tr = tr.link) {  System.***out***.print(tr.data + " ");  }  }  System.***out***.println();  }  } |