**Algorithms**

**Laboratory Task-7**

**Submission Deadline** – As announced in the class

**Submission Guidelines**-

* Rename the file to your id only. If your id is 18-XXXXX-1, then the file name must be 18-XXXXX-1.docx.
* Must submit within the given deadline in VUES to the section named Lab Tak-1
* Must include resources for all the section named ‘Code’ and ‘Output (screenshot)’ in the table.

|  |
| --- |
| Question-1 – **Longest Common Sub-sequence (LCS)** |
| **Pseudocode**  // m is the length of the first String  // n is the length of the second String  for(i = 0; i less than or equal to m; i++) {  for(j = 0; j less than or equal to n; j++) {  if ( i == 0) { // Entries on first row  mat[i][j] = 0;  }  else if ( j == 0) { // Entries on first column  mat[i][j] = 0;  }  else if (first[i] == second[j]) { // Same character on both strings  mat[i][j] = mat[i-1][j-1]+1;  }  else if (first[i] != second[j]) { // Different characters on both strings  mat[i][j] = max((mat[i-1][j]),(mat[i][j-1]));  }  } // end for j  } // end for i |
| **Code** |
| **Output (Screenshot)** |

|  |
| --- |
| Question-2 – **Implement Breadth First Search (BFS)** |
| **Pseudocode**  BFS (G, s) //Where G is the graph and s is the source node  let Q be queue.  Q.enqueue( s ) //Inserting s in queue until all its neighbour vertices are marked.  mark s as visited.  while ( Q is not empty)  //Removing that vertex from queue,whose neighbour will be visited now  v = Q.dequeue( )  //processing all the neighbours of v  for all neighbours w of v in Graph G  if w is not visited  Q.enqueue( w ) //Stores w in Q to further visit its neighbour  mark w as visited. |
| **Code** |
| **Output (Screenshot)** |

|  |
| --- |
| Question-3 – **Implement Depth First Search (DFS)** |
| **Pseudocode**  DFS-iterative (G, s): //Where G is graph and s is source vertex  let S be stack  S.push( s ) //Inserting s in stack  mark s as visited.  while ( S is not empty):  //Pop a vertex from stack to visit next  v = S.top( )  S.pop( )  //Push all the neighbours of v in stack that are not visited  for all neighbours w of v in Graph G:  if w is not visited :  S.push( w )  mark w as visited |
| **Code** |
| **Output (Screenshot)** |