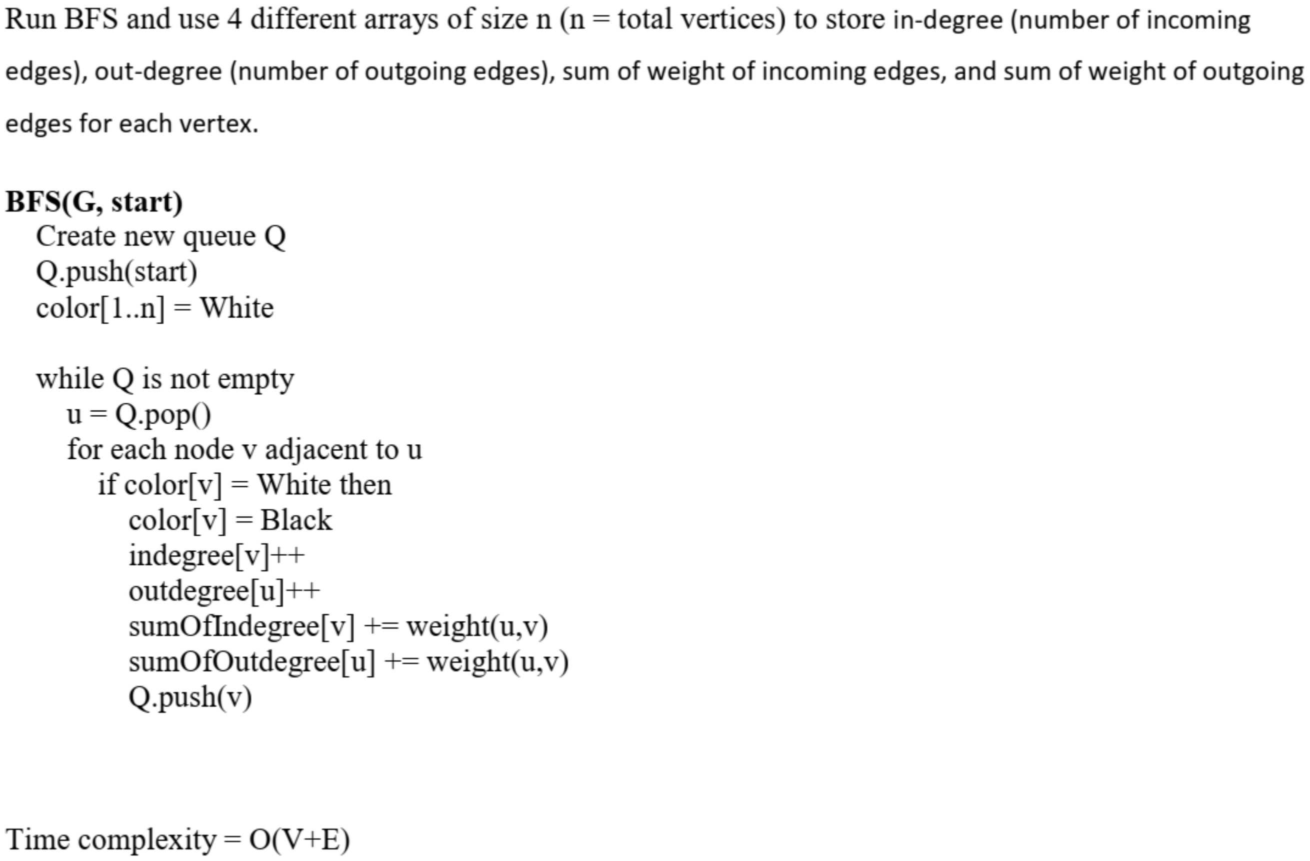
Name: ____________________ Reg #: ____________________ Section: ______________

## National University of Computer and Emerging Sciences, Lahore Campus

| | | | |
|---|---|---|---|
| **Course:** | **Design and Analysis of Algorithms** | **Course Code:** | **CS2009** |
| **Program:** | **BS(Computer Science)** | **Semester:** | **Fall 2022** |
| **Duration:** | **20 Minutes** | **Total Marks:** | **10** |
| **Paper Date:** | **28-Nov-2022** | **Weight** | **%** |
| **Section:** | **5C** | **Page(s):** | **2** |
| **Exam:** | **Quiz 4** | | |

**Q1)** Given a weighted directed graph in adjacency list representation, write an efficient algorithm to calculate in-degree (number of incoming edges), out-degree (number of outgoing edges), sum of weight of incoming edges, and sum of weight of outgoing edges for each vertex of the graph. Analyze time complexity of your algorithm. [10 Marks]

**Solution**



Run BFS and use 4 different arrays of size n (n = total vertices) to store in-degree (number of incoming edges), out-degree (number of outgoing edges), sum of weight of incoming edges, and sum of weight of outgoing edges for each vertex.

```
BFS(G, start)
  Create new queue Q
  Q.push(start)
  color[1..n] = White

  while Q is not empty
    u = Q.pop()
    for each node v adjacent to u
      if color[v] = White then
        color[v] = Black
        indegree[v]++
        outdegree[u]++
        sumOfIndegree[v] += weight(u,v)
        sumOfOutdegree[u] += weight(u,v)
        Q.push(v)
```

Time complexity = O(V+E)