**Department of Computer Science**

**Forman Christian College University**

**COMP360: Introduction to AI**

**Spring 2024**

**Lab 2**



|  |  |  |
| --- | --- | --- |
| **Task 1 (10)** | **Task 2 (10)** | **Total (20)** |
|  |  |  |

**BFS Implementations**

Hafsah Shahbaz 251684784

This lab is aimed to give you the opportunity to practice and observe the practical implementation of Graph and Breadth-First-Search (FBS).

**Important Instructions:**

* Lab Manual is self-explanatory so read through this lab manual before starting the lab.
* You can use any of the **Programming Language** that you think works better for you in this Lab (Python is preferred).
* Remember Copy and Pasting of the code is prohibited and is not a good practice.
* Every question asked in the viva will have marks, so make sure you understand what you write and should be able to explain the code.
* No marks will be given without viva even if you have submitted the lab.
* Late submission will not be accepted.

**Expected Deliverable:**

* **The Code Files:** 2 File having the implementations of Graph and BFS.

**Submission Instruction:**

You need to submit the following files along with the specified file names on moodle.

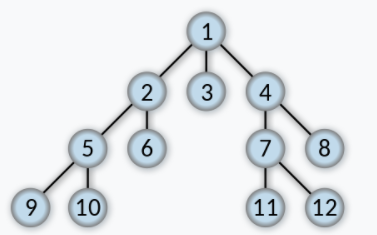
* Graph file 🡪 “lab2\_graph\_name\_roll.py”
* BFS file 🡪 “lab2\_BFS\_name\_roll.py”

**Experimental Tasks:**

You need to perform the following tasks in order to qualify this Lab.

Use the provided python files.

**Task 1:** Observe the following **Tree.** Consider that node having “**1”** is the root node.



**Task 2:** Write the code for **BFS,** try the above tree node’s numbers as inputsand write down the results you observed.

def breadth\_first\_search(*graph*, *start\_node*):

    """

    The function should take in the graph defined along with the

    start node and print out the path according

    to the Breadth First Search Algorithm.

    NOTE: print the path

    :params graph: (Graph) defined graph

    :params start\_node: (String) starting node from graph

    :return : None

    """

    visited = set()

    queue = deque([*start\_node*])

    while queue:

        current = queue.popleft()

        if current not in visited:

            print(current, *end*=" ")

            visited.add(current)

            neighbours = *graph*.neighbours(current)

            for neighbour in neighbours:

                queue.append(neighbour)

***Note: This is an individual Lab and Groups are not allowed.***