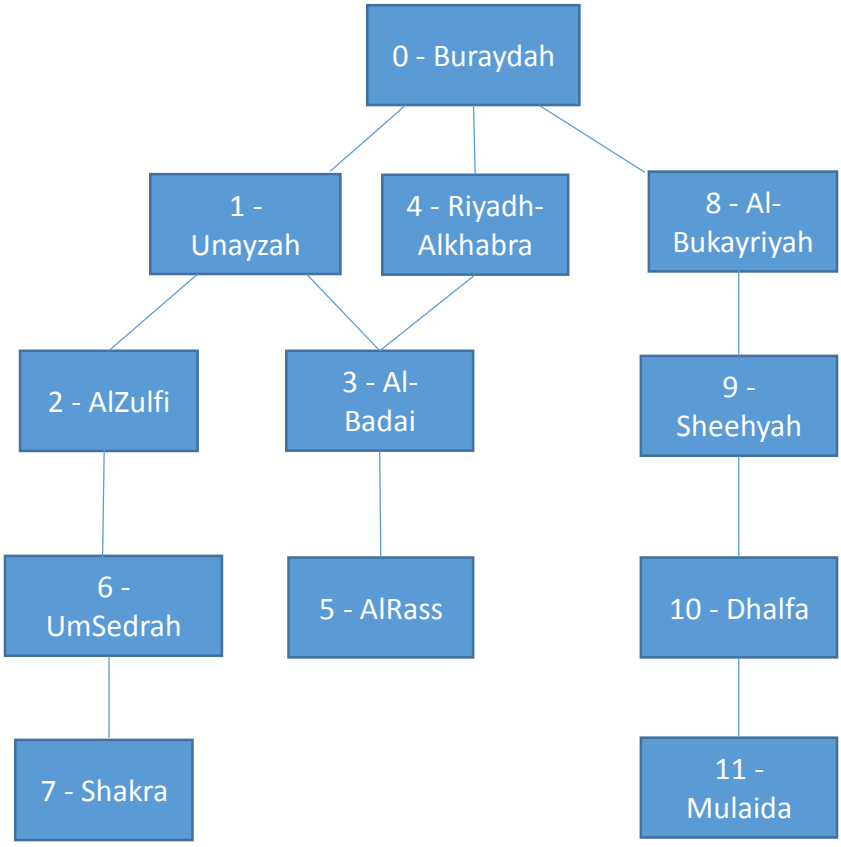
Lab 3: Traversing a graph using FIFO (First In First Out) and LIFO (Last In First Out) Strategies

Introduction

This lab is about traversing a graph using FIFO (First In First Out) or LIFO (Last In First Out) fashion. FIFO is related to the queues and the first element added in the linked list is the first one to be removed. Similarly LIFO is related to the stack, whereby the last element added is the first one to be removed. The example given in this lab is from the local context i.e. Qassim Region as shown in the figure below. The same example would be used in many of the subsequent labs which makes this lab more important.



Description and Tasks

The source code in Java is provided in Appendix A. You need to run the code and understand how different functions have been implemented. Add screenshots with different inputs. You need

to see the difference between FIFO and LIFO with respect to the provided graph. Moreover, try to remove certain edge (s) and observe the behavior of FIFO and LIFO.

Appendix A - Source Code for FIFO and LIFO Implementation (Lab 2):

```
import java.util.*;
public class TRA {
   int N;                                  // number of vertices in the graph
   boolean[ ][ ] G;                        // the graph as an adjacency matrix
                                           // G[i][j] is true if there is an edge from i to j
   int currPos;

   TRA(int size,int loc)
   {
      N=size;
      currPos=loc;
      setupGraph();
      System.out.println("----------Traverse----------");
      System.out.println();
   }

   void setupGraph()
   {
      // set up a graph with 8 vertices that looks like:

      G=new boolean[N][N];

      G[0][1]=G[1][0]=true;                // notice that for each edge    G[i][j] == G[j][i]
      G[0][8]=G[8][0]=true;                // means the graph is undirected
      G[0][4]=G[4][0]=true;
      G[1][2]=G[2][1]=true;
      G[1][3]=G[3][1]=true;
      G[2][6]=G[6][2]=true;
      G[3][4]=G[4][3]=true;
      G[3][5]=G[5][3]=true;
      G[6][7]=G[7][6]=true;
      G[8][9]=G[9][8]=true;
      G[9][10]=G[10][9]=true;
```

```
    G[10][11]=G[11][10]=true;
  }

  void TRA()
  {
    System.out.println("The current location is: "+retCity(currPos)+"\n");
    System.out.println("Moving to the next destination: \n..[0] manually\n..[1] First In First Out
(FIFO)\n..[2] Last In First Out (LIFO)]\n..[-1] Stop");
    Scanner inp=new Scanner(System.in);
    int ch=inp.nextInt();

    switch(ch)                          //The user chooses the way to traverse the graph
    {
      case -1:                          //Stop
      System.out.println("The program's been stopped");
      break;

      case 0:
      List<Integer> list=new ArrayList<Integer>();
      for(int i=0;i<N;i++)
      if (G[currPos][i]==true)
      list.add(i);

      System.out.println("\nChoose one of the following connected cities: ");
      for(int i=0;i<list.size();i++)
        System.out.println("["+i+"]"+retCity(list.get(i))+"\t");
      System.out.println("\nInput: ");
      ch=inp.nextInt();


      if(ch<0 || ch>=list.size())
      {
        System.out.println("The choice is not correct,try again");
        TRA();
      }
      currPos=list.get(ch);
      TRA();
```

```
        break;

        case 1:                                 // FIFO
        String addToQ="[(Front) ";
        Queue<Integer> Q=new LinkedList<Integer>(); // A queue to process nodes
        for(int i=0;i<N;i++)
        if(G[currPos][i]==true){
           Q.offer(i);//The connected cities are added to the queue
        addToQ+=retCity(i)+"\t";}
        addToQ+="]\n";

        System.out.println(addToQ);
        System.out.println(retCity(Q.peek())+" is the destination\n");
        currPos=Q.peek();
        TRA();
        break;

        case 2:                                 // LIFO
        String addToSta="";
        Stack<Integer> sta=new Stack<Integer>();
        addToSta+="[";
        for(int i=0;i<N;i++)
        if(G[currPos][i]==true){
           sta.push(i);//The connected cities are added to the statck
        addToSta+=retCity(i)+"\t";}
        addToSta+="top=>]";
        System.out.println(addToSta);
        currPos=sta.peek();

        TRA();
        break;
    }
}

public static void main(String[] args)
{
   int cityChoice;
```

```
        Scanner inp=new Scanner(System.in);
        System.out.println("\n\nChoose a city number to start with: \n");
        for(int i=0;i<12;i++)//shows the list of cities
        System.out.println(retCity(i)+ " city["+i+"]");
        System.out.print("\nInput: ");
        cityChoice=inp.nextInt();//User's choice of the list

        if(cityChoice<0 ||cityChoice>=12)//the user choice is not included in the list
        {
           System.out.println("Mistake,run the program again");
           System.exit(0);
        }

        TRA traGraph=new TRA(12,cityChoice);
        traGraph.TRA();
    }

    public static String retCity(int i) // the function returns city name, according to its index in V
array
    {
        if(i==0)
        return "Buraydah";
        else if(i==1)
        return "Unayzah";
        else if(i==2)
        return "AlZulfi";
        else if(i==3)
        return "Al-Badai";
        else if(i==4)
        return "Riyadh-Alkhabra";
        else if(i==5)
        return "AlRass";
        else if(i==6)
        return "UmSedrah";
        else if(i==7)
        return "Shakra";
        else if(i==8)
```

```
        return "Al-Bukayriyah";
        else if(i==9)
        return "Sheehyah";
        else if(i==10)
        return "Dhalfa";
        else  return "Mulida";

    }
}
```