TREE – LAB 06

EC 4070

DATA STRUCTURES AND ALGORITHMS

NAME : **WIJAYAWARDHANA W.A.H.A.**

REGISTRATION NO. : **2019/E/166**

SEMESTER : **SEMESTER 04**

DATE ASSIGNED : **24 MARCH 2022**

01.

Code:-

```
public class CityDatabase {
   double[][] cityDetails = new double[20][3];
   String[][] cityNameArray = new String[20][2];
   int arraySize = 3;
   double longitude;
   double latitude;
   String cityName;
   int root;
   int idNumber = 1;

   /**
    * setCityDataDetails method use for set city details to the 2D array.
    * @param cityName
    * @param latitude
    * @param longitude
    * @param index
    */
   public void setCityDataDetails(String cityName , double latitude , double longitude , int
index)
   {
      if(root == 0)
      {
         cityDetails[1][0] = idNumber;
         cityDetails[1][1] = latitude;
         cityDetails[1][2] = longitude;
         cityNameArray[1][0] = String.valueOf(idNumber);
         cityNameArray[1][1] = cityName;
         idNumber++;
         System.out.println(cityName + " added.");
      }
      else
      {
         this.cityName = cityName;
         this.latitude = latitude;
         this.longitude = longitude;
         cityDetails[index][0] = idNumber;
         cityDetails[index][1] = latitude;
         cityDetails[index][2] = longitude;
         cityNameArray[index][0] = String.valueOf(idNumber);
         cityNameArray[index][1] = cityName;
         idNumber++;
      }
```

```
}

/**
 * Insertion method use for add elements into the tree.
 * @param cityName
 * @param latitude
 * @param longitude
 */

public void insertion(String cityName , double latitude , double longitude)
{
   this.cityName = cityName;
   this.latitude = latitude;
   this.longitude = longitude;
   root = 1;
   findArrayIndex(root);
}

/**
 * This alphabeticalOrder method use to set city names in alphabetical order.
 * @param city01
 * @param city02
 * @return
 */
public boolean alphabeticalOrder(String city01 , String city02)
{
   return city01.compareTo(city02)>0;
}

/**
 * This method use to find the array index.
 * @param i
 */
public void findArrayIndex(int i)
{
   boolean city02High = alphabeticalOrder(cityNameArray[i][1],cityName);
   if(city02High == true)
   {
      if(cityNameArray[2*root][1] == null)
      {
         cityNameArray[2*root][0] = String.valueOf(2*root);
         cityNameArray[2*root][1] = cityName;
         System.out.println(cityName + " added.");
         setCityDataDetails(cityName,longitude,latitude,2*root);
         return;
      }

      else
      {
```

```
                findArrayIndex((2*root));
            }
        }
        else
        {
            if(cityNameArray[2*root+1][1] == null)
            {
                cityNameArray[2*root+1][0] = String.valueOf(2*(root+1));
                cityNameArray[2*root+1][1] = cityName;
                System.out.println(cityName + " added.");
                setCityDataDetails(cityName,longitude,latitude,2*root+1);
                return;
            }

            else
            {
                findArrayIndex((2*root+1));
            }
        }
    }

    /**
     * printDetails method use to print the details of the array.
     */
    public void printDetails()
    {
        System.out.println("Print details.");
        for (int i = 1; i < arraySize; i++)
        {
            for (int j = i + 1; j < arraySize; j++)
            {
                if(cityNameArray[i][1].compareTo(cityNameArray[j][1]) > 0)
                {
                    String temp = cityNameArray[i][1];
                    String temp2 = cityNameArray[i][0];
                    cityNameArray[i][1] = cityNameArray[j][1];
                    cityNameArray[i][0] = cityNameArray[j][0];
                    cityNameArray[j][1] = temp;
                    cityNameArray[j][0] = temp2;
                }
            }

        }
        for(int i =0; i<cityNameArray.length;i++)
        {
            if(cityNameArray[i][1] != null)
                System.out.println(cityNameArray[i][1]);
        }
```

```
}

/**
 * descendingOrderPrint method use to print tree in descending order.
 */

public void descendingOrderPrint()
{
   for(int i = cityNameArray.length-1; i>=0;i--)
   {
      if(cityNameArray[i][1] != null)
         System.out.println(cityNameArray[i][1]);
   }
}

/**
 * deleteItem method use to delete element from the tree.
 * @param cityNameToDelete
 */
public void deleteItem(String cityNameToDelete)
{
   int i =0;
   for(; i<arraySize; i++)
   {
      if(cityNameArray[i][1] == cityNameToDelete)
      {
         System.out.println(cityNameArray[i][1] + " deleted.");
         cityNameArray[i][0] = null;
         cityNameArray[i][1] = null;
         arraySize--;
      }
   }
}

/**
 * searchCityName method use to search the element using name of the city.
 * @param searchCityName
 */
public void searchName(String searchCityName)
{
   boolean isFound = false;
   for(int i =0; i<arraySize; i++)
   {
      if(cityNameArray[i][1] == searchCityName)
      {
         System.out.println(searchCityName+ " founded.");
         isFound = true;
      }
   }
```

```
        if(isFound == false)
        {
            System.out.println(searchCityName+ " do not found.");
        }
    }

    /**
     * calculateDistance method use to get the distance less than the given value.
     * @param distance01
     */
    public void calculateDistance(double distance01)
    {
        for(int i = 0; i<cityDetails.length; i++)
        {
            if((cityDetails[i][1] < distance01)&&((cityDetails[i][1] >0)||(cityDetails[i][1] <0)))
            {
                System.out.println("City less than "+distance01+ " : "+cityNameArray[i][1]);
            }
        }


    }

    /**
     * main method to run the written methods.
     * @param args
     */

    public static void main(String[] args) {
        CityDatabase newObject = new CityDatabase();
        newObject.setCityDataDetails("Colombo",6.927079,79.861244,0);
        newObject.insertion("Chicago",41.881832,-87.623177);
        newObject.insertion("Sydney" , -33.865143 ,151.20990);
        newObject.calculateDistance(72.495865);
        newObject.printDetails();
        newObject.deleteItem("Colombo");
        newObject.printDetails();
        newObject.searchName("Chicago");
        newObject.searchName("Kandy");
        System.out.println("Descending order : ");
        newObject.descendingOrderPrint();
        newObject.printDetails();

    }
}
```

Output:

```
Run:  CityDatabase
C:\Users\User\.jdks\openjdk-17.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Educat
Colombo added.
Chicago added.
Sydney added.
City less than 72.495865 : Colombo
City less than 72.495865 : Chicago
Print details.
Chicago
Colombo
Sydney
Colombo deleted.
Print details.
Chicago
Sydney
Chicago founded.
Kandy do not found.
Descending order :
Sydney
Chicago
Print details.
Chicago
Sydney

Process finished with exit code 0
```

FIGURE 01 - OUTPUT