## Laboratory Exercise 8

# Searching and traversal on lists, trees and graphs

1. Design and implement a searching algorithm which search for a *name* from a list of student records (linked structure). Let each record contain the following fields: `name, age` and `address`.

2. Implement the following algorithms for a binary search tree:

```
1.InsertOneNode(NewItem, TreeNode)
----------------------------------
  begin {algorithm}
    if TreeNode is Empty then
       place NewItem in TreeNode
    else
       if NewItem precedes item in TreeNode then
          InsertOneNode(NewItem, LeftChild of TreeNode)
       else
          InsertOneNode(NewItem, RightChild of TreeNode)
  end {algorithm};

2.DisplayAll(TreeNode)
----------------------
  begin {algorithm}
    if TreeNode is not Empty then
      begin
        DisplayAll(LeftChild of TreeNode);
        display item in TreeNode;
        DisplayAll(RightChild of TreeNode)
      end
  end {algorithm};
```

3. Using array approach, write a class which constructs an adjacent matrix of a given graph.

4. Define an array of linked list. Implementing the adjacent list of a given graph, write a class which constructs an adjacent list of a given graph.

5. Implement the *depth-first* and the *breadth-first* searching algorithms provided below.

   (a) Depth-first searching algorithm:

```
method DFS(v:VertexType);
variables
  w:VertexType;

begin
  visit and mark v;
  while there is a unmarked vertex w
       adjacent to v do
    DFS(w)
end
```

(b) Breadth-first searching algorithm:

```
method BFS(adjacencyList:HeaderList;
             v:VertexType);
variable
  Q:queue;
  w,x:VertexType;

begin
  Initialize(Q);
  visit and mark v; Enqueue(Q,v);
  while not Empty(Q) do
    begin
      Dequeue(Q,x);
      for each unmarked vertex w adjacent to x do
        begin
          visit and mark w;
          Enqueue(Q,w)
        end {for}
    end {while}
end
```