Objective:

- The purpose of this lab is to have an algorithmic understanding of some basic tree view operations shown in windows explorer (desktop browser).
- It focuses on the creation of trees, which doesn't have any limit on degree as it is in windows explorer.

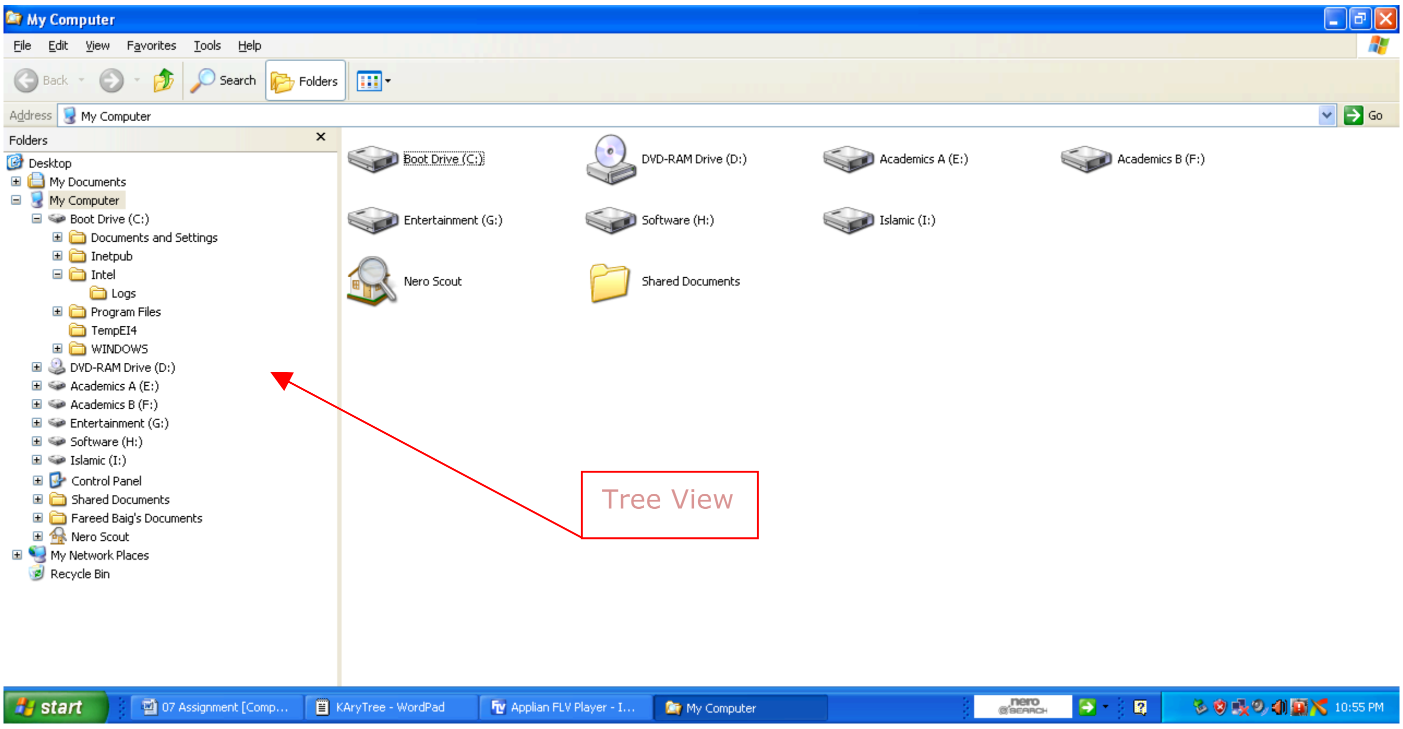


Figure-A

Operations/Algorithms needs to be implemented:

1. Designing an algorithm to find the size in bytes of a particular file or folder.
2. Algorithm for giving an expanded view of a particular folder/drive as shown in the following diagram.

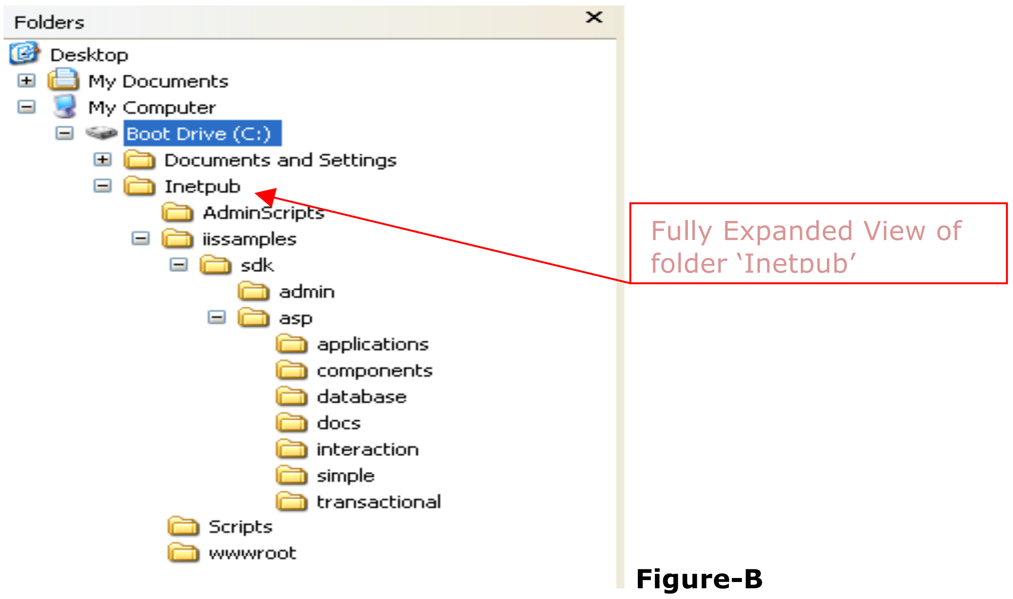


Figure-B

3. Algorithm for displaying a particular folder in fully expanded Parenthesized view.
4. Check whether a given path is valid or not.

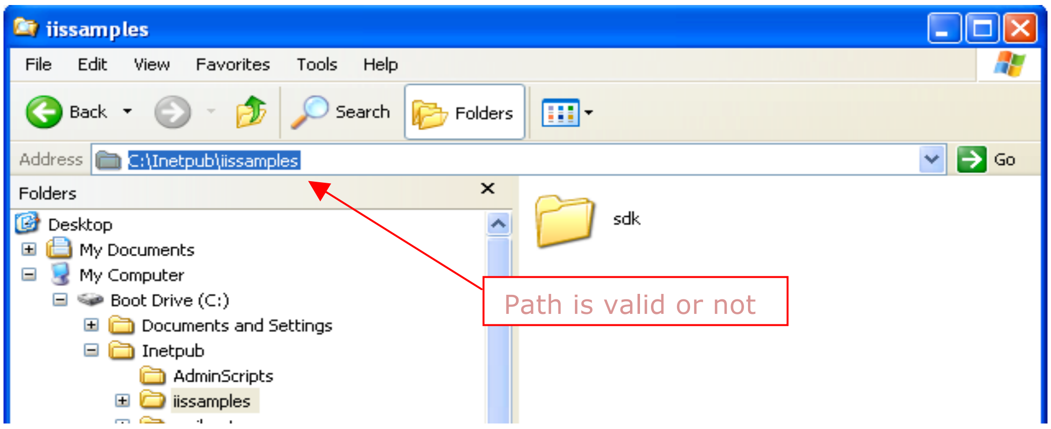


Figure-C

5. Copy/Cut a folder/file from source to destination

Classes/Structures needed for the above cited operations to be done are as follows:

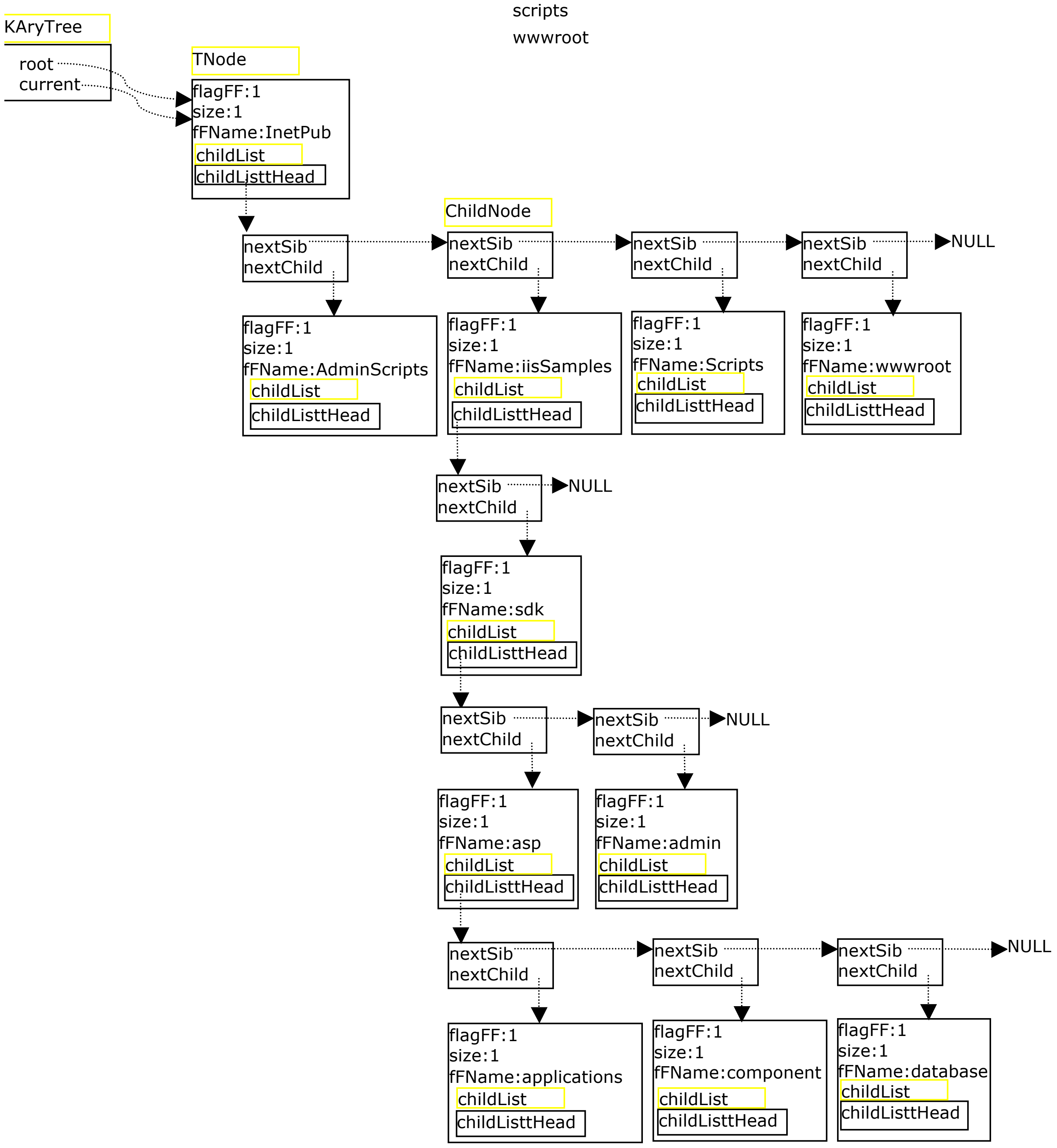
Explanation of each of the 5 operations needed is given at the end.

```
class ChildNode
{
public:
      ChildNode *nextSib;
      TNode * nextChild;
};

class ChildList
{
public:
      ChildNode *childListHead;
};
class TNode
{
public:
      int flagFF; //flagFF shows that a particular node is a file or folder 0 means
                  //file 1 means folder
      int size; //represents the size of folder or file: we take an assumption that a
                  //folder will always take 1KB and a file size will be given by user
                  //in KB.
                  // In other words an empty or non-empty folder will always take 1K
                  //and leaf nodes (Files) will take the size as given by the user
      string fFName; represents file/folder name.

      ChildList childList;
};
class KAryTree // This tree don't have any limit on degree.
{
private:
      TNode * root;      // points to the root of the node
      TNode * current;   // points to the current/selected file/folder on which the
                         //operation is to done.
public:
};
```

Graphical View of the classes/structures in memory:

Following diagram shows the diagrammatic view of folder 'InetPub' in Figure-B according to our classes/structures.

```
-InetPub
        AdminScripts
        -iisSamples
                -sdk
                        -asp
                                applications
                                component
                                database
                        admin
        scripts
        wwwroot
```



Operations/Algorithms needed to be implemented:

1. Designing an algorithm to find the size in bytes of a particular file or folder.
Prototype: int KAryTree::findSize(string path);

Find the size of "C:\DSA\F06\Assignment\"
So for this you have to find the size of folder 'Assignment' and this folder will become the current folder.
If you don't pass the path then the size of current folder will be calculated.

2. Algorithm for giving an expanded view of a particular folder/drive as shown in the following diagram.

Prototype: void KAryTree::expandedView(string path);
According to figure-B, the fully expanded view of Path "C:\IntePub\" will be displayed on console as follows:

```
-InetPub
        AdminScripts
        -iisSamples
                -sdk
                        admin
                        -asp
                                applications
                                components
                                databases
                                docs
                                interaction
                                samples
                                transactional
        scripts
        wwwroot
```

Note: if path not given then the fully expanded view of current node should be displayed.

3. Algorithm for displaying a particular folder in fully expanded Parenthesized view (discussed in Lab-10).

Prototype: void KAryTree::parenthsizedView(string path);

4. Check whether a given path exists.

Prototype: int KAryTree::isFound(string path);

As according to figure-B, the path "C:\DSA" does not exists so it returns 0 otherwise returns 1.

5. Copy/Cut a folder/file from source to destination

Prototype: void KAryTree::copyPaste(string sourcePath, string destPath, int cp);

Copy the particular folder/file from source to destination path where cp==1 mean copy and cp==0 means cut operation.

Volunteer Task

6. Generate the KAryTree Object from the given parenthesize KAry Tree expression.

Prototype: KAryTree KAryTree::generateTree(string parenthesizeExp);