Markdown Preview

First, calculate and store the size of each subtree using tree traversal.

Next, traverse both the trees A and B simultaneously, let the current node in node in tree B be nodeB. Following 4 cases arise :

```
1. if( nodeA==NULL and nodeB == NULL )
     return
2. if( nodeA==NULL and nodeB != NULL )
     numberofadditions+= size of the subtree rooted at nodeB
3. if( nodeA!=NULL and nodeB == NULL )
      numberofdeletions+= size of the subtree rooted at nodeA
4. if( nodeA!=NULL and nodeB!= NULL )
     if( value at nodeA != value at nodeB )
      numberofedits+=1
```

Note: It was observed that many students made mistakes such as improper ir input or accesing the fields of a NULL pointer during recursive functions whic fault / no output.sive function which resulted in segmentation fault / no outp

Markdown Preview

First, calculate and store the size of each subtree using tree traversal.

Next, traverse both the trees A and B simultaneously, let the current node in node in tree B be nodeB. Following 4 cases arise :

```
1. if( nodeA==NULL and nodeB == NULL )
     return
2. if( nodeA==NULL and nodeB != NULL )
     numberofadditions+= size of the subtree rooted at nodeB
3. if( nodeA!=NULL and nodeB == NULL )
      numberofdeletions+= size of the subtree rooted at nodeA
4. if( nodeA!=NULL and nodeB!= NULL )
     if( value at nodeA != value at nodeB )
      numberofedits+=1
```

Note: It was observed that many students made mistakes such as improper ir input or accesing the fields of a NULL pointer during recursive functions whic fault / no output.sive function which resulted in segmentation fault / no outp