

# Inlusto Practical Examinations for DSA

Sharmila R

211082

3<sup>rd</sup> CSE B.

1. Finding intersecting node if we merge two singly linked list.

1.1. Algorithm for finding the merging point:

Step 1: First we need to create nodes for singly linked list.

Step 2: And then create method for `getIntersectionNode()`.

Step 3: Get count of the nodes in the first and second list, let count them as  $c_1$  and  $c_2$ .

Step 4: Get the difference of counts if  $(c_1 > c_2)$ , then  $d = (c_1 - c_2)$ ; else  $d = (c_2 - c_1)$ .

Step-5: Now traverse the bigger list from the lists equal no. of nodes that from here onwards both the lists have equal no. of nodes.

Step 6: Then we have to traverse both the lists in parallel till we come across a common node.

Step 7: Then we can find the merging point of the 2 linked lists.

```
102
103     newNode = new Node();
104     newNode->data = 5;
105     head2->next->next = newNode;
106
107     newNode = new Node();
108     newNode->data = 7;
109     head1->next = newNode;
110     head2->next->next->next = newNode;
111
112     newNode = new Node();
113     newNode->data = 4;
114     head1->next->next = newNode;
115
116     head1->next->next->next = NULL;
117
118     cout << "The node of intersection is " << getIntersectionNode(head1, head2);
119 }
```

input

The node of intersection is 7

...Program finished with exit code 0  
Press ENTER to exit console.

1.2.Can we solve this using the sorting technique?

**Yes,**we can solve this problem using the sorting technique.We have to first create an array and store all the address of the nodes in this array.And then sort the array.For each element in the second list,we have to search fpr the address in the array.If we find a same memory address,then that is the merging point of the 2 points.

3. Can we solve it using hash tables?

**Yes,**we can solve this using problem using hash tables.

1.4 Can we use stacks for solving?

**Yes,**we can solve this using problem using stacks.We have to create 2 differeent stacks and push the elements in respective stacks.We have to pop the elements from the stack at once,till both lists are merged(we will getthe same value from both stacks).If both stacks returns different values then the last popped element is the merging point of the list.

1.5. Is there any other way of solving this ?

**Yes**, we can solve this problem using other methods too.

Example: Brute-Force method.

1.6. Can we improve the complexity for?

**Yes**, we can improve the complexity for this problem.

2. Give an algorithm for finding the size of a binary tree.

Step 1: 1.Size() recursively calculates the size of a tree.

Step 2: The size of the tree is calculated by using the formula: Size of a tree = Size of left subtree + 1 + Size of right subtree.

Step 3: call the method Size().

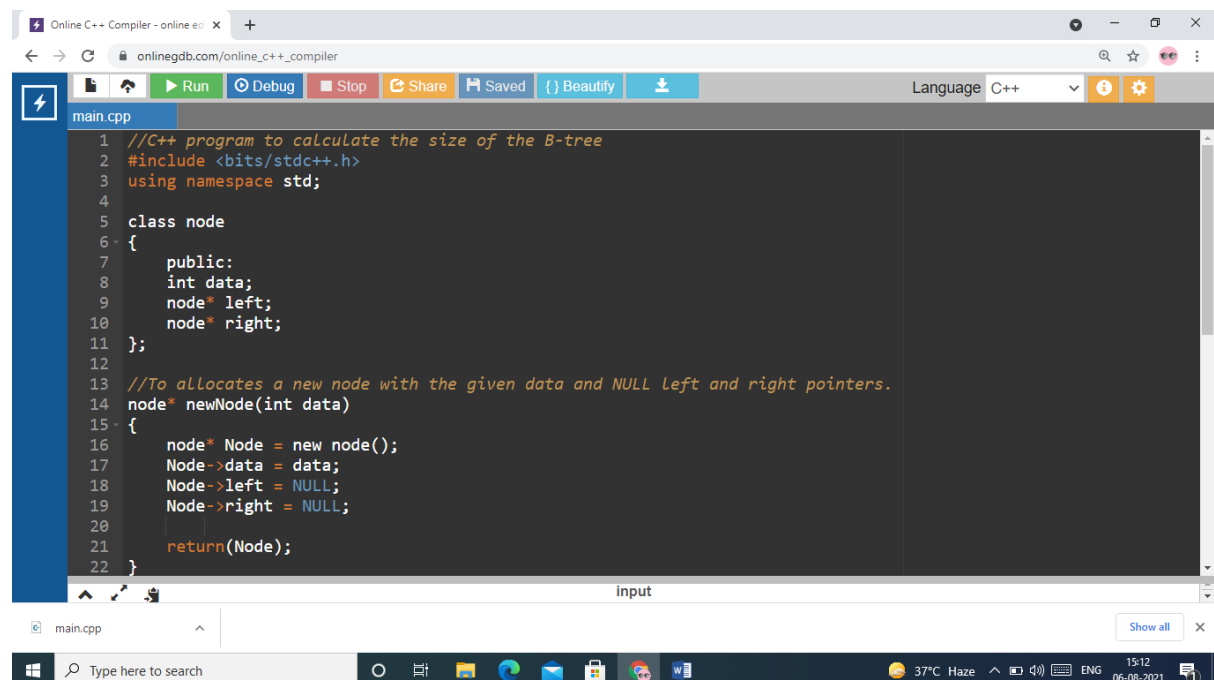
i) If tree is empty it will return 0.

ii) Else it gets the size of left and right subtree recursively.

Then calculate the size of the tree.

Step 4: Then we have to give input for B-tree.

Step 5: Then it will print the size of B-tree.



The screenshot shows a web browser window with an online C++ compiler. The code in the editor is as follows:

```
1 //C++ program to calculate the size of the B-tree
2 #include <bits/stdc++.h>
3 using namespace std;
4
5 class node
6 {
7     public:
8     int data;
9     node* left;
10    node* right;
11 };
12
13 //To allocate a new node with the given data and NULL left and right pointers.
14 node* newNode(int data)
15 {
16     node* Node = new node();
17     Node->data = data;
18     Node->left = NULL;
19     Node->right = NULL;
20
21     return(Node);
22 }
```

The browser's address bar shows `onlinegdb.com/online_c++_compiler`. The bottom of the image shows a Windows taskbar with the date and time as 15:12 on 06-08-2021.

The screenshot shows a web browser window with the URL `onlinegdb.com/online_c++_compiler`. The browser's address bar and tabs are visible at the top. Below the browser window is a toolbar with buttons for Run, Debug, Stop, Share, Saved, and Beautify. The main area of the browser displays a C++ code editor with the following code:

```
24  /* Computes the number of nodes in a tree. */
25  int size(node* node)
26  {
27      if (node == NULL)
28          return 0;
29      else
30          return(size(node->left) + 1 + size(node->right));
31  }
32
33  //Main
34  int main()
35  {
36      node *root = newNode(6);
37      root->left = newNode(3);
38      root->right = newNode(9);
39      root->left->left = newNode(1);
40      root->left->right = newNode(5);
41      root->right->left = newNode(7);
42      root->right->right = newNode(11);
43
44      cout << "Size of the tree is " << size(root);
45      return 0;

```

Below the code editor is an input field. At the bottom of the browser window, the Windows taskbar is visible, showing the search bar, task view button, and several application icons. The system tray on the right shows the temperature (37°C), weather (Haze), and the date and time (15:12, 06-08-2021).

2.a) How will you solve it without recursion?

If we don't use recursive, we need a data structure to store the tree traversal, we will use queue .