Convert string to tree.

![Diagram

Description automatically generated]()

Here dash ‘-‘means negative.

1st edge case is no input and 2nd is 4) like invalid input.

Here we only deal with valid input.

Use stack->encounter number put it in the stack.

Runtime will be linear on pass->O(n)

Code:

![Text, letter

Description automatically generated]()edge case string is empty.

Code in c++

// C++ program for the above approach

#include <bits/stdc++.h>

using namespace std;

// Build a tree node having left and

// right pointers set to null initially

struct Node {

Node\* left;

Node\* right;

int data;

// Constructor to set the data of

// the newly created tree node

Node(int element)

{

data = element;

this->left = nullptr;

this->right = nullptr;

}

};

// Utility function to print

// preorder traversal of the tree

void preorder(Node\* root)

{

if (!root)

return;

cout << root->data << " ";

preorder(root->left);

preorder(root->right);

}

// Function to construct a

// tree using bracket notation

Node\* constructTree(string s)

{

// First character is the root of the tree

Node\* root = new Node(s[0] - '0');

// Stack used to store the

// previous root elements

stack<Node\*> stk;

// Iterate over remaining characters

for (int i = 1; i < s.length(); i++) {

// If current character is '('

if (s[i] == '(') {

// Push root into stack

stk.push(root);

}

// If current character is ')'

else if (s[i] == ')') {

// Make root the top most

// element in the stack

root = stk.top();

// Remove the top node

stk.pop();

}

// If current character is a number

else {

// If left is null, then put the new

// node to the left and move to the

// left of the root

if (root->left == nullptr) {

Node\* left = new Node(s[i] - '0');

root->left = left;

root = root->left;

}

// Otherwise, if right is null, then

// put the new node to the right and

// move to the right of the root

else if (root->right == nullptr) {

Node\* right = new Node(s[i] - '0');

root->right = right;

root = root->right;

}

}

}

// Return the root

return root;

}

// Driver code

int main()

{

// Input

string s = "4(2(3)(1))(6(5))";

// Function calls

Node\* root = constructTree(s);

preorder(root);

return 0;

}