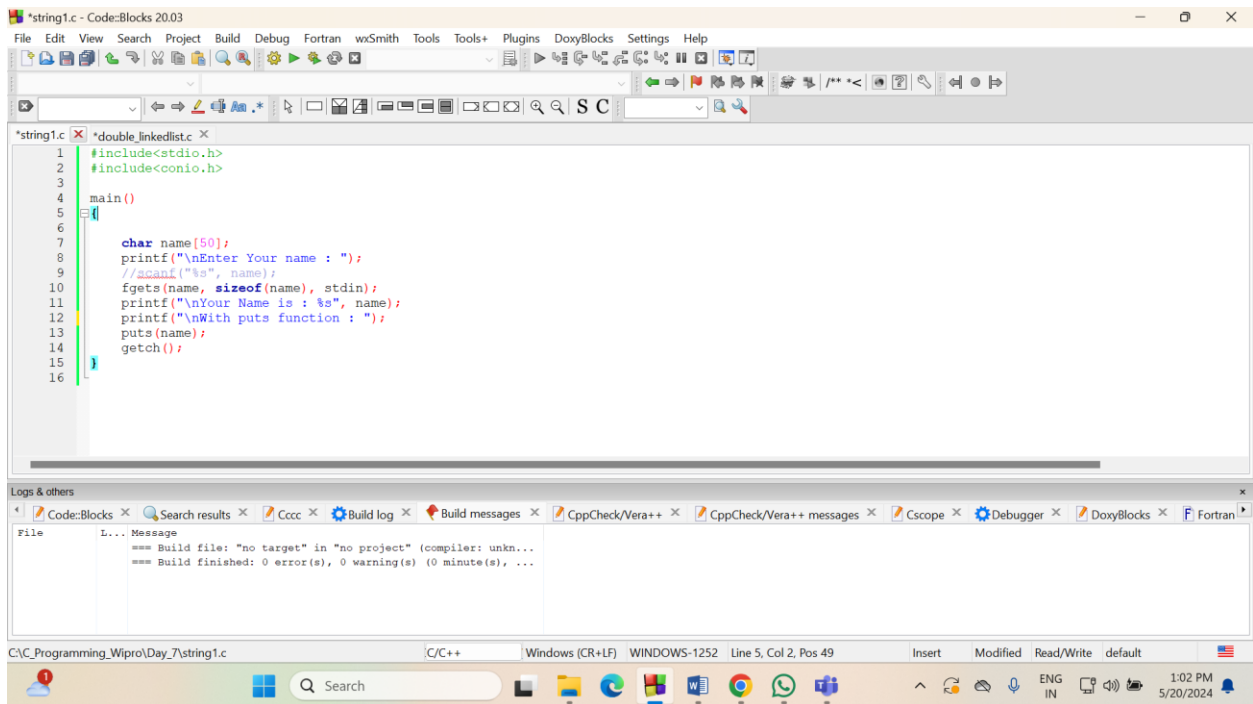


## Day 7 C Program

String:



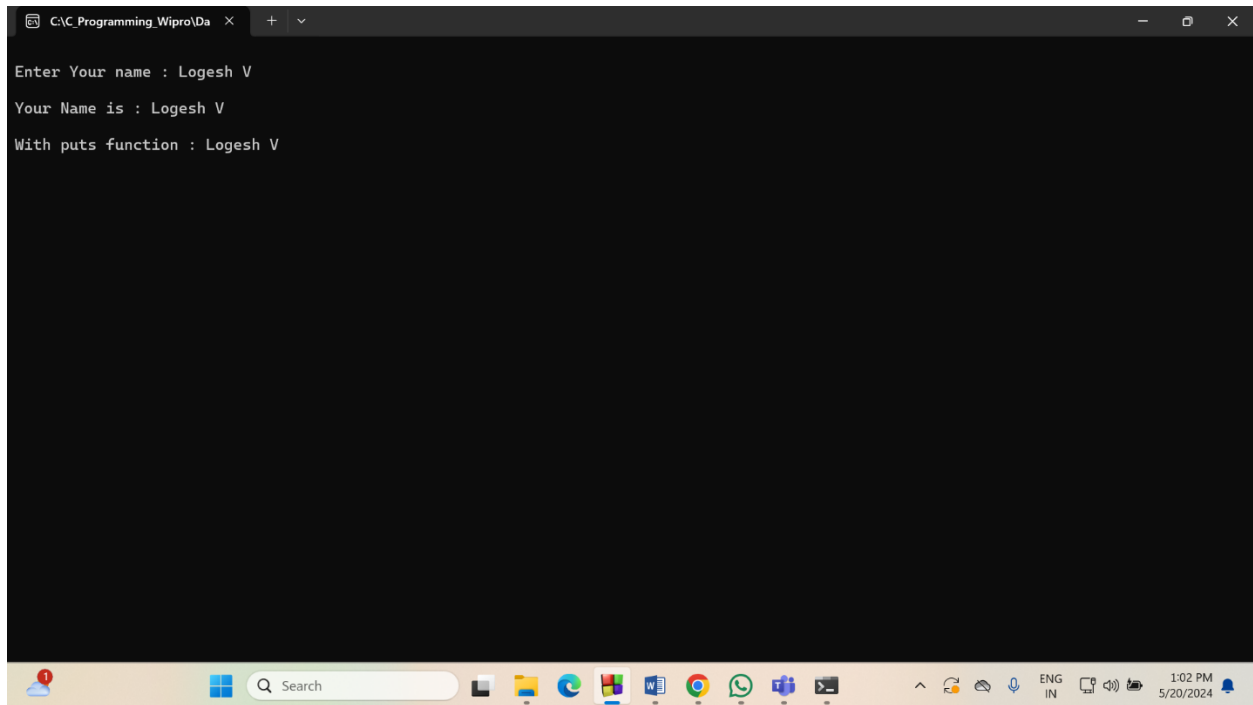
The screenshot shows the Code::Blocks IDE with a C program named `*string1.c`. The program includes `<stdio.h>` and `<conio.h>`. It defines a `main()` function where a character array `name` of size 50 is declared. The program prompts the user to enter their name using `printf`, reads the input using `fgets`, displays the input using `printf`, and then displays the input using the `puts` function. The `getch()` function is used to pause the program before exiting.

```
1 #include<stdio.h>
2 #include<conio.h>
3
4 main()
5 {
6
7     char name[50];
8     printf("\nEnter Your name : ");
9     //scanf("%s", name);
10    fgets(name, sizeof(name), stdin);
11    printf("\nYour Name is : %s", name);
12    printf("\nWith puts function : ");
13    puts(name);
14    getch();
15 }
16
```

The bottom panel shows the 'Logs & others' window with the following message:

```
File      L... Message
===== Build file: "no target" in "no project" (compiler: unkn...
===== Build finished: 0 error(s), 0 warning(s) (0 minute(s), ...
```

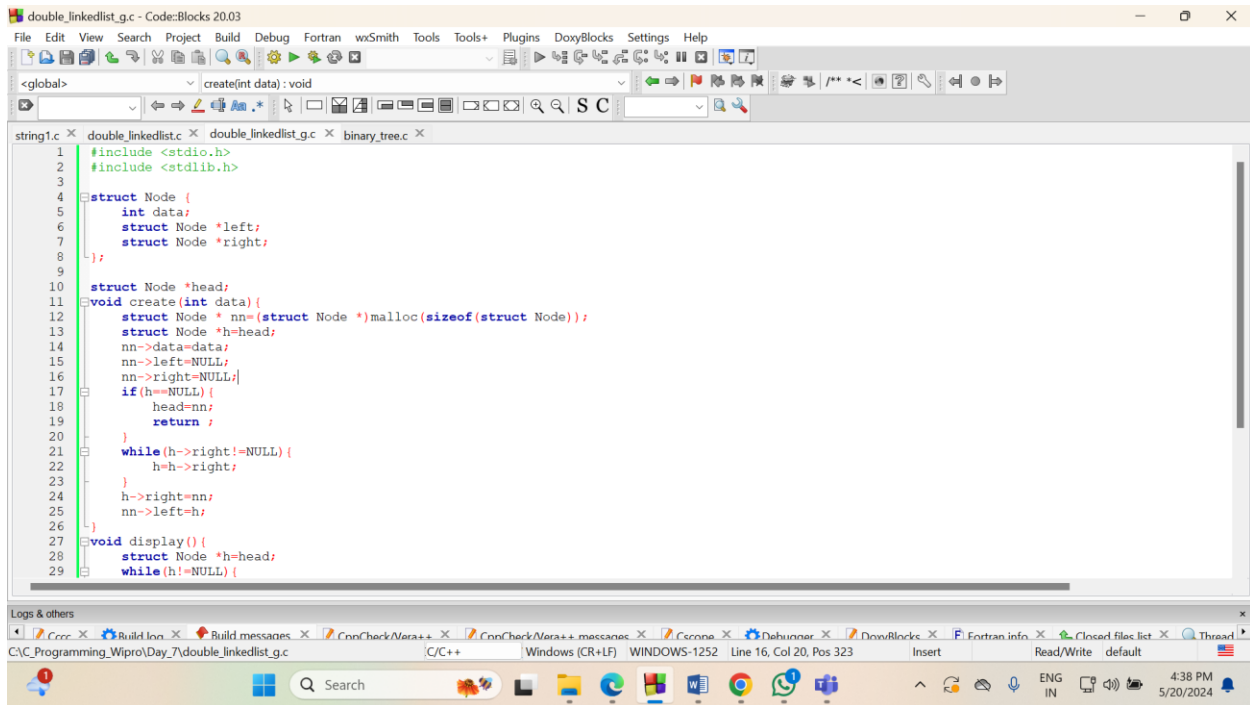
Output:



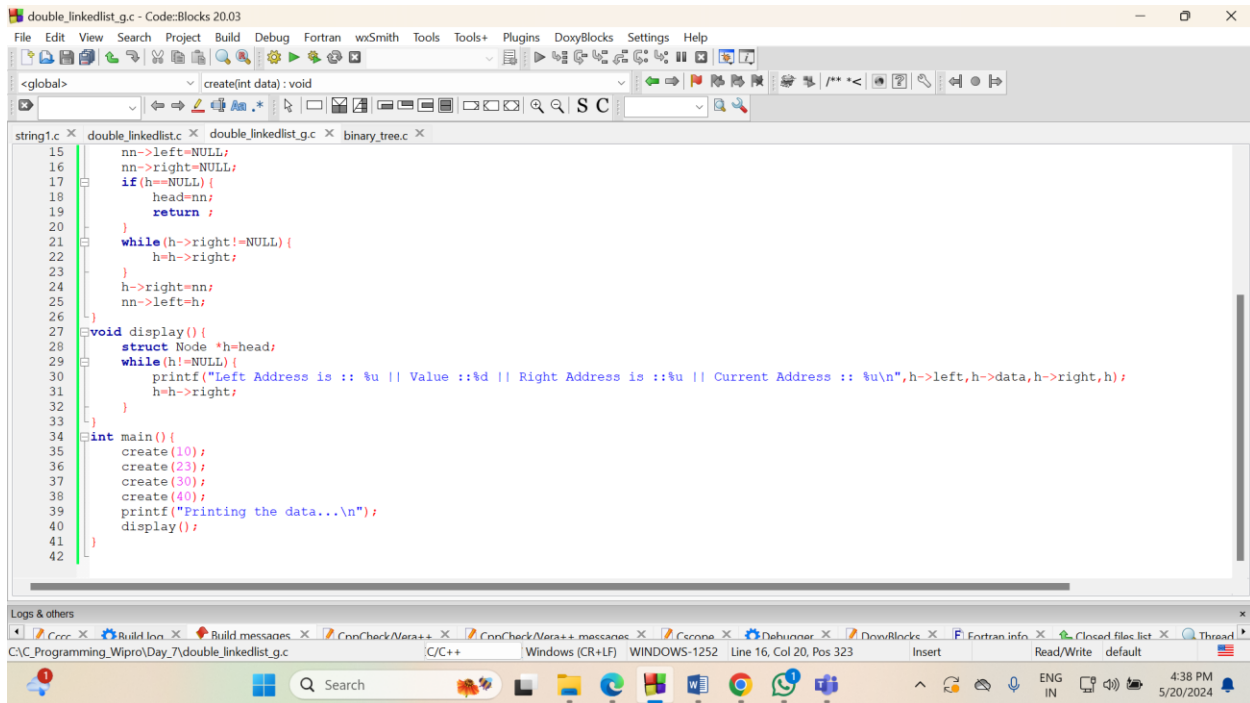
The screenshot shows a terminal window with the output of the C program. The user has entered 'Logesh V' as their name. The program outputs the name using both `printf` and the `puts` function.

```
Enter Your name : Logesh V
Your Name is : Logesh V
With puts function : Logesh V
```

## Doubly linkedlist:



```
double_linkedlist.g.c - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
<global> create(int data): void
string1.c double_linkedlist.c double_linkedlist.g.c binary_tree.c
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 struct Node {
5     int data;
6     struct Node *left;
7     struct Node *right;
8 };
9
10 struct Node *head;
11 void create(int data){
12     struct Node * nn=(struct Node *)malloc(sizeof(struct Node));
13     struct Node *h=head;
14     nn->data=data;
15     nn->left=NULL;
16     nn->right=NULL;
17     if(h==NULL){
18         head=nn;
19         return ;
20     }
21     while(h->right!=NULL){
22         h=h->right;
23     }
24     h->right=nn;
25     nn->left=h;
26 }
27 void display(){
28     struct Node *h=head;
29     while(h!=NULL){
```



```
double_linkedlist.g.c - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
<global> create(int data): void
string1.c double_linkedlist.c double_linkedlist.g.c binary_tree.c
15 nn->left=NULL;
16 nn->right=NULL;
17 if(h==NULL){
18     head=nn;
19     return ;
20 }
21 while(h->right!=NULL){
22     h=h->right;
23 }
24 h->right=nn;
25 nn->left=h;
26 }
27 void display(){
28     struct Node *h=head;
29     while(h!=NULL){
30         printf("Left Address is :: %u || Value :: %d || Right Address is :: %u || Current Address :: %u\n",h->left,h->data,h->right,h);
31         h=h->right;
32     }
33 }
34 int main(){
35     create(10);
36     create(23);
37     create(30);
38     create(40);
39     printf("Printing the data...\n");
40     display();
41 }
42 }
```

Output:

```
C:\C_Programming_Wipro\Da x + v
Printing the data...
Left Address is :: 0 || Value ::10 || Right Address is ::11932752 || Current Address :: 11932720
Left Address is :: 11932720 || Value ::23 || Right Address is ::11932784 || Current Address :: 11932752
Left Address is :: 11932752 || Value ::30 || Right Address is ::11932816 || Current Address :: 11932784
Left Address is :: 11932784 || Value ::40 || Right Address is ::0 || Current Address :: 11932816

Process returned 0 (0x0)   execution time : 0.152 s
Press any key to continue.
```

Binary Tree:

```
binary_tree.c - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
<global> insert() : Node
string1.c x double_linkedlist.c x double_linkedlist.g.c x binary_tree.c x
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 struct Node {
5     int a;
6     struct Node *left;
7     struct Node *right;
8 };
9
10 struct Node *root = NULL;
11
12 struct Node* insert() {
13     int data;
14     struct Node *nn=(struct Node*)malloc(sizeof(struct Node));
15     printf("Enter Data [-1 for start inserting left or right]\n");
16     scanf("%d", &data);
17     if(data == -1)
18         return 0;
19     nn->a = data;
20     printf("Enter Left Node Data ");
21     nn->left=insert();
22     printf("Enter Right Node Data ");
23     nn->right=insert();
24     return nn;
25 }
26
27 void preorder(struct Node *root) {
28     if (root == NULL) {
29         return;
```

binary\_tree.c - Code::Blocks 20.03

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

<global> insert(): Node

```
string1.c x double_linkedlist.c x double_linkedlist.g.c x binary_tree.c x
27 void preorder(struct Node *root) {
28     if (root == NULL) {
29         return;
30     }
31     printf("%d ", root->a);
32     preorder(root->left);
33     preorder(root->right);
34 }
35 void inorder(struct Node *root) {
36     if (root == NULL) {
37         return;
38     }
39     inorder(root->left);
40     printf("%d ", root->a);
41     inorder(root->right);
42 }
43 void postorder(struct Node *root) {
44     if (root == NULL) {
45         return;
46     }
47     postorder(root->left);
48     postorder(root->right);
49     printf("%d ", root->a);
50 }
51 }
52 int main() {
53     root = insert();
54     printf("Printing the data in the list[preOrder Traversal]\n");
55 }
```

Logs & others

C:\C\_Programming\_Wipro\Day\_7\binary\_tree.c C/C++ Windows (CR+LF) WINDOWS-1252 Line 15, Col 31, Pos 288 Insert Read/Write default 4:35 PM 5/20/2024

binary\_tree.c - Code::Blocks 20.03

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

<global> insert(): Node

```
string1.c x double_linkedlist.c x double_linkedlist.g.c x binary_tree.c x
39     inorder(root->left);
40     printf("%d ", root->a);
41     inorder(root->right);
42 }
43 void postorder(struct Node *root) {
44     if (root == NULL) {
45         return;
46     }
47     postorder(root->left);
48     postorder(root->right);
49     printf("%d ", root->a);
50 }
51 }
52 int main() {
53     root = insert();
54     printf("Printing the data in the list[preOrder Traversal]\n");
55     preorder(root);
56
57     printf("\nPrinting the data in the list[inOrder Traversal]\n");
58     inorder(root);
59
60     printf("\nPrinting the data in the list[postOrder Traversal]\n");
61     postorder(root);
62
63     return 0;
64 }
65
66 }
```

Logs & others

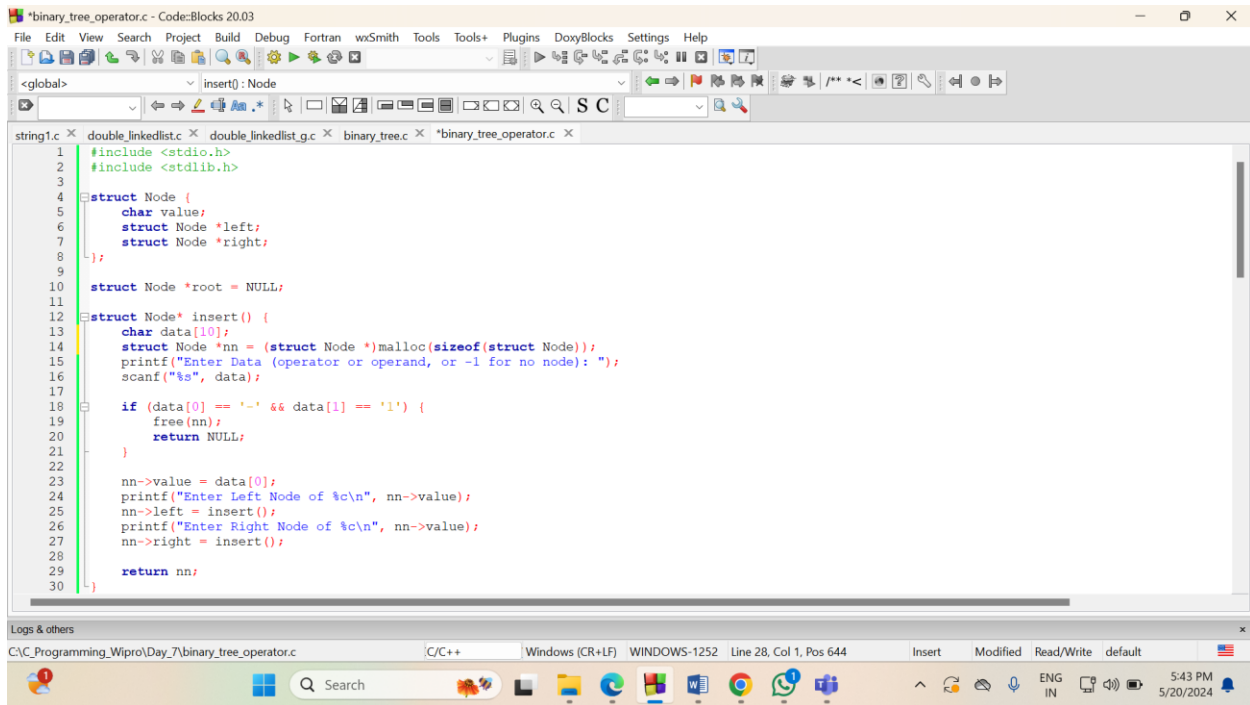
C:\C\_Programming\_Wipro\Day\_7\binary\_tree.c C/C++ Windows (CR+LF) WINDOWS-1252 Line 15, Col 31, Pos 288 Insert Read/Write default 4:35 PM 5/20/2024

Output:

```
C:\C_Programming_Wipro\Da  X + v
Enter Data [-1 for start inserting left or right]
1
Enter Left Node Data Enter Data [-1 for start inserting left or right]
2
Enter Left Node Data Enter Data [-1 for start inserting left or right]
3
Enter Left Node Data Enter Data [-1 for start inserting left or right]
-1
Enter Right Node Data Enter Data [-1 for start inserting left or right]
-1
Enter Right Node Data Enter Data [-1 for start inserting left or right]
4
Enter Left Node Data Enter Data [-1 for start inserting left or right]
-1
Enter Right Node Data Enter Data [-1 for start inserting left or right]
-1
Enter Right Node Data Enter Data [-1 for start inserting left or right]
5
Enter Left Node Data Enter Data [-1 for start inserting left or right]
6
Enter Left Node Data Enter Data [-1 for start inserting left or right]
-1
Enter Right Node Data Enter Data [-1 for start inserting left or right]
-1
Enter Right Node Data Enter Data [-1 for start inserting left or right]
7
Enter Left Node Data Enter Data [-1 for start inserting left or right]
-1
Enter Right Node Data Enter Data [-1 for start inserting left or right]
-1
Printing the data in the list[preOrder Traversal]
1 2 3 4 5 6 7
Printing the data in the list[inOrder Traversal]
```

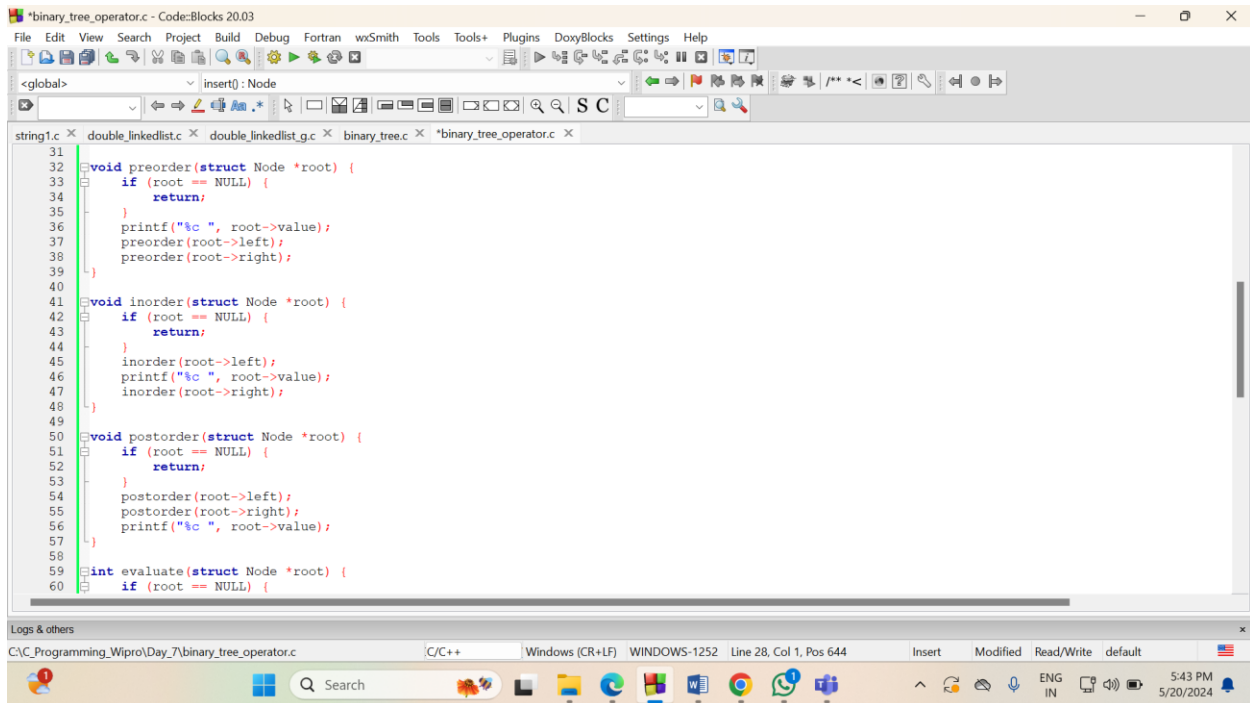
```
C:\C_Programming_Wipro\Da  X + v
Enter Left Node Data Enter Data [-1 for start inserting left or right]
-1
Enter Right Node Data Enter Data [-1 for start inserting left or right]
-1
Enter Right Node Data Enter Data [-1 for start inserting left or right]
4
Enter Left Node Data Enter Data [-1 for start inserting left or right]
-1
Enter Right Node Data Enter Data [-1 for start inserting left or right]
-1
Enter Right Node Data Enter Data [-1 for start inserting left or right]
5
Enter Left Node Data Enter Data [-1 for start inserting left or right]
6
Enter Left Node Data Enter Data [-1 for start inserting left or right]
-1
Enter Right Node Data Enter Data [-1 for start inserting left or right]
-1
Enter Right Node Data Enter Data [-1 for start inserting left or right]
7
Enter Left Node Data Enter Data [-1 for start inserting left or right]
-1
Enter Right Node Data Enter Data [-1 for start inserting left or right]
-1
Printing the data in the list[preOrder Traversal]
1 2 3 4 5 6 7
Printing the data in the list[inOrder Traversal]
3 2 4 1 6 5 7
Printing the data in the list[postOrder Traversal]
3 4 2 6 7 5 1
Process returned 0 (0x0)    execution time : 34.392 s
Press any key to continue.
```

## Binary tree with operators and string:



The screenshot shows the Code::Blocks IDE with the file `*binary_tree_operator.c` open. The code defines a `Node` structure with `value`, `left`, and `right` pointers. It also includes a `insert()` function that takes a string of data and inserts it into a binary tree. The function checks for a root node, allocates a new node, and recursively inserts the left and right children based on the input string.

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 struct Node {
5     char value;
6     struct Node *left;
7     struct Node *right;
8 };
9
10 struct Node *root = NULL;
11
12 struct Node* insert() {
13     char data[10];
14     struct Node *nn = (struct Node *)malloc(sizeof(struct Node));
15     printf("Enter Data (operator or operand, or -1 for no node): ");
16     scanf("%s", data);
17
18     if (data[0] == '-' && data[1] == '1') {
19         free(nn);
20         return NULL;
21     }
22
23     nn->value = data[0];
24     printf("Enter Left Node of %c\n", nn->value);
25     nn->left = insert();
26     printf("Enter Right Node of %c\n", nn->value);
27     nn->right = insert();
28
29     return nn;
30 }
```



The screenshot shows the Code::Blocks IDE with the file `*binary_tree_operator.c` open. The code defines three traversal functions: `preorder`, `inorder`, and `postorder`. Each function takes a `struct Node *` as input and prints the value of the node and its children in the specified order. The `evaluate` function is also shown, which takes a `struct Node *` as input and returns the value of the node.

```
31
32 void preorder(struct Node *root) {
33     if (root == NULL) {
34         return;
35     }
36     printf("%c ", root->value);
37     preorder(root->left);
38     preorder(root->right);
39 }
40
41 void inorder(struct Node *root) {
42     if (root == NULL) {
43         return;
44     }
45     inorder(root->left);
46     printf("%c ", root->value);
47     inorder(root->right);
48 }
49
50 void postorder(struct Node *root) {
51     if (root == NULL) {
52         return;
53     }
54     postorder(root->left);
55     postorder(root->right);
56     printf("%c ", root->value);
57 }
58
59 int evaluate(struct Node *root) {
60     if (root == NULL) {
```

\*binary\_tree\_operator.c - Code::Blocks 20.03

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

<global> insert(): Node

```
string1.c x double_linkedlist.c x double_linkedlist.g.c x binary_tree.c x *binary_tree_operator.c x
61     return 0;
62 }
63
64 if (root->left == NULL && root->right == NULL) {
65     return root->value - '0';
66 }
67
68
69 int left_val = evaluate(root->left);
70 int right_val = evaluate(root->right);
71
72 switch (root->value) {
73     case '+': return left_val + right_val;
74     case '-': return left_val - right_val;
75     case '*': return left_val * right_val;
76     case '/': return left_val / right_val;
77 }
78
79 return 0;
80 }
81
82 int main() {
83     printf("Building the expression tree...\n");
84     root = insert();
85
86     printf("PreOrder Traversal: ");
87     preorder(root);
88     printf("\n");
89
90     printf("InOrder Traversal: ");
```

Logs and others

C:\C\_Programming\_Wipro\Day\_7\binary\_tree\_operator.c | C/C++ | Windows (CR+LF) | WINDOWS-1252 | Line 28, Col 1, Pos 644 | Insert | Modified | Read/Write | default | 5:43 PM 5/20/2024

\*binary\_tree\_operator.c - Code::Blocks 20.03

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

<global> insert(): Node

```
string1.c x double_linkedlist.c x double_linkedlist.g.c x binary_tree.c x *binary_tree_operator.c x
76     case '/': return left_val / right_val;
77 }
78
79 return 0;
80 }
81
82 int main() {
83     printf("Building the expression tree...\n");
84     root = insert();
85
86     printf("PreOrder Traversal: ");
87     preorder(root);
88     printf("\n");
89
90     printf("InOrder Traversal: ");
91     inorder(root);
92     printf("\n");
93
94     printf("PostOrder Traversal: ");
95     postorder(root);
96     printf("\n");
97
98     printf("Evaluating the expression tree...\n");
99     int result = evaluate(root);
100    printf("Result of the expression: %d\n", result);
101
102    return 0;
103 }
104 }
```

Logs and others

C:\C\_Programming\_Wipro\Day\_7\binary\_tree\_operator.c | C/C++ | Windows (CR+LF) | WINDOWS-1252 | Line 28, Col 1, Pos 644 | Insert | Modified | Read/Write | default | 5:44 PM 5/20/2024

Output:

```
C:\C_Programming_Wipro\Da x + v
Building the expression tree...
Enter Data (operator or operand, or -1 for no node): A
Enter Left Node of A
Enter Data (operator or operand, or -1 for no node): +
Enter Left Node of +
Enter Data (operator or operand, or -1 for no node): B
Enter Left Node of B
Enter Data (operator or operand, or -1 for no node): -1
Enter Right Node of B
Enter Data (operator or operand, or -1 for no node): -1
Enter Right Node of +
Enter Data (operator or operand, or -1 for no node): C
Enter Left Node of C
Enter Data (operator or operand, or -1 for no node): -1
Enter Right Node of C
Enter Data (operator or operand, or -1 for no node): -1
Enter Right Node of A
Enter Data (operator or operand, or -1 for no node): -
Enter Left Node of -
Enter Data (operator or operand, or -1 for no node): D
Enter Left Node of D
Enter Data (operator or operand, or -1 for no node): -1
Enter Right Node of D
Enter Data (operator or operand, or -1 for no node): -1
Enter Right Node of -
Enter Data (operator or operand, or -1 for no node): E
Enter Left Node of E
Enter Data (operator or operand, or -1 for no node): -1
Enter Right Node of E
Enter Data (operator or operand, or -1 for no node): -1
PreOrder Traversal: A + B C - D E
InOrder Traversal: B + C A D - E
PostOrder Traversal: B C + D E - A
```