**Formation of tree using infix and postfix expression:**

1. #include<iostream>
2. using namespace std;
3. struct node
4. {
5. int a;
6. node \*l,\*r;
7. }\*root='\0',\*head;
8. int preind=0;
9. int sea(int [],int,int,int);
10. node \*build(int in[20],int post[20],int instrt,int inend,int poststrt,int postend)
11. {
12. node \*cnode=new node;
13. cnode->a=post[postend];
14. cnode->l='\0';
15. cnode->r='\0';
16. if(preind==0)
17. head=cnode;
18. preind++;
19. if(instrt==inend)
20. return cnode;
21. else
22. {
23. int index=sea(in,instrt,inend,post[postend]);
24. if(instrt<index)
25. cnode->l=build(in,post,instrt,index-1,poststrt,poststrt+index-(instrt+1));
26. if(inend>index)
27. cnode->r=build(in,post,index+1,inend,poststrt+index-instrt,postend-1);
28. }
29. return cnode;
30. }
31. int sea(int in[20],int first,int last,int val)
32. {
33. for(int i=first;i<=last;++i)
34. if(val==in[i])
35. return i;
36. }
37. void preorder(node \*root)
38. {
39. cout<<root->a<<" ";
40. if(root->l!='\0')
41. preorder(root->l);
42. if(root->r!='\0')
43. preorder(root->r);
44. }
45. int main()
46. {
47. int n,in[20],post[20];
48. cout<<"\n enter no of nodes:";
49. cin>>n;
50. cout<<"\n enter inorder:";
51. for(int i=0;i<n;++i)
52. cin>>in[i];
53. cout<<"\n enter postorder:";
54. for(int i=0;i<n;++i)
55. cin>>post[i];
56. root=build(in,post,0,n-1,0,n-1);
57. cout<<"\n preorder:";
58. preorder(head);
59. return 0;
60. }

**Formation of tree using infix and prefix expression:**

1. #include<iostream>
2. using namespace std;
3. struct node
4. {
5. int a;
6. node \*l,\*r;
7. }\*root='\0',\*head;
8. int preind=0;
9. int sea(int [],int,int,int);
10. node \*build(int in[20],int pre[20],int instrt,int inend)
11. {
12. node \*cnode=new node;
13. cnode->a=pre[preind];
14. cnode->l='\0';
15. cnode->r='\0';
16. if(preind==0)
17. head=cnode;
18. preind++;
19. if(instrt==inend)
20. return cnode;
21. else
22. {
23. int index=sea(in,instrt,inend,pre[preind-1]);
24. if(instrt<index)
25. cnode->l=build(in,pre,instrt,index-1);
26. if(inend>index)
27. cnode->r=build(in,pre,index+1,inend);
28. }
29. return cnode;
30. }
31. int sea(int in[20],int first,int last,int val)
32. {
33. for(int i=first;i<=last;++i)
34. if(val==in[i])
35. return i;
36. }
37. void postorder(node \*root)
38. {
39. if(root->l!='\0')
40. postorder(root->l);
41. if(root->r!='\0')
42. postorder(root->r);
43. cout<<root->a<<" ";
44. }
45. int main()
46. {
47. int n,in[20],pre[20];
48. cout<<"\n enter no of nodes:";
49. cin>>n;
50. cout<<"\n enter inorder:";
51. for(int i=0;i<n;++i)
52. cin>>in[i];
53. cout<<"\n enter preorder:";
54. for(int i=0;i<n;++i)
55. cin>>pre[i];
56. root=build(in,pre,0,n-1);
57. cout<<"\n postorder:";
58. postorder(head);
59. return 0;
60. }

**Deletion in binary search tree:**

1. #include<iostream>
2. using namespace std;
3. struct node
4. {
5. int a;
6. node \*l,\*r;
7. }\*root='\0',\*temp='\0',\*head='\0',\*p,\*y,\*p1;
8. void form(node \*);
9. void del(int,node\*);
10. void preorder(node\*);
11. int main()
12. {
13. int x;
14. char ch;
15. node \*n;
16. n=new node;
17. n->r='\0';
18. n->l='\0';
19. cout<<"\n enter root:";
20. cin>>n->a;
21. root=n;
22. head=root;
23. do
24. {
25. root=head;
26. cout<<"\n continue?";
27. cin>>ch;
28. if(ch=='y')
29. {
30. node \*n;
31. n=new node;
32. n->r='\0';
33. n->l='\0';
34. cout<<"\n enter data:";
35. cin>>n->a;
36. form(n);
37. }
38. }while(ch!='n');
39. cout<<"\n preorder is:";
40. root=head;
41. preorder(root);
42. root=head;
43. cout<<"\n enter the node data value to be deleted:";
44. cin>>x;
45. del(x,root);
46. root=head;
47. cout<<"\n preorder is:";
48. preorder(root);
49. return 0;
50. }
51. void form(node \*n)
52. {
53. if(n->a > root->a)
54. {
55. if(root->r=='\0')
56. root->r=n;
57. else
58. {
59. root=root->r;
60. form(n);
61. }
62. }
63. else if(n->a <=root->a)
64. {
65. if(root->l=='\0')
66. root->l=n;
67. else
68. {
69. root=root->l;
70. form(n);
71. }
72. }
73. }
74. void del(int x,node \*t)
75. {
76. while(t->a!=x)
77. {
78. p=t;
79. if(x<=t->a)
80. t=t->l;
81. else
82. t=t->r;
83. }
84. if(t->l=='\0'&& t->r=='\0')
85. {
86. if(p->l==t)
87. p->l='\0';
88. else
89. p->r='\0';
90. }
91. if((t->l=='\0'&& t->r!='\0') || (t->r=='\0'&& t->l!='\0'))
92. {
93. if(p->l==t && t->l=='\0')
94. p->l=t->r;
95. else if(p->r==t && t->l=='\0')
96. p->r=t->r;
97. else if(p->l==t && t->r=='\0')
98. p->l=t->l;
99. else
100. p->r=t->l;
101. }
102. if(t->l!='\0'&& t->r!='\0')
103. {
104. y=t->l;
105. while(y->r!='\0')
106. {
107. p1=y;
108. y=y->r;
109. }
110. t->a=y->a;
111. if(y->l=='\0'&& y->r=='\0')
112. {
113. if(p1->l==y)
114. p1->l='\0';
115. else
116. p1->r='\0';
117. }
118. if((y->l=='\0'&& y->r!='\0') || (y->r=='\0'&& y->l!='\0'))
119. {
120. if(p1->l==y && y->l=='\0')
121. p1->l=y->r;
122. else if(p1->r==y && y->l=='\0')
123. p1->r=y->r;
124. else if(p1->l==y && y->r=='\0')
125. p1->l=y->l;
126. else
127. p1->r=y->l;
128. }
129. }
130. }
131. void preorder(node \*root)
132. {
133. cout<<root->a<<" ";
134. if(root->l!='\0')
135. preorder(root->l);
136. if(root->r!='\0')
137. preorder(root->r);
138. }