**AVL tree insertion:**

1. #include<iostream>
2. using namespace std;
3. struct node
4. {
5. int data;
6. node \*l,\*r;
7. int h;
8. }\*root='\0';
9. int getbal(node \*);
10. int height(node \*n)
11. {
12. if(n=='\0')
13. return 0;
14. else
15. return n->h;
16. }
17. int maximum(int a,int b)
18. {
19. return (a>b)?a:b;
20. }
21. node \*newnode(int data)
22. {
23. node \*n=new node;
24. n->data=data;
25. n->l='\0';
26. n->r='\0';
27. n->h=1;
28. return n;
29. }
30. node \*rightrotate(node \*y)
31. {
32. node \*x=y->l;
33. node \*t=x->r;
34. x->r=y;
35. y->l=t;
36. y->h=maximum(height(y->l),height(y->r))+1;
37. x->h=maximum(height(x->l),height(x->r))+1;
38. return x;
39. }
40. node \*leftrotate(node \*x)
41. {
42. node \*y=x->r;
43. node \*t=y->l;
44. y->l=x;
45. x->r=t;
46. x->h=maximum(height(x->l),height(x->r))+1;
47. y->h=maximum(height(y->l),height(y->r))+1;
48. return y;
49. }
50. node \*insertion(node \*n,int data)
51. {
52. if(n=='\0')
53. return (newnode(data));
54. if(data<n->data)
55. n->l=insertion(n->l,data);
56. else
57. n->r=insertion(n->r,data);
58. n->h=maximum(height(n->l),height(n->r))+1;
59. int bal;
60. bal=getbal(n);
61. if(bal>1 && data<(n->l->data))
62. return rightrotate(n);
63. if(bal<-1 && data>(n->r->data))
64. return leftrotate(n);
65. if(bal>1 && data>(n->l->data))
66. {
67. n->l=leftrotate(n->l);
68. return rightrotate(n);
69. }
70. if(bal<-1 && data<(n->r->data))
71. {
72. n->r=rightrotate(n->r);
73. return leftrotate(n);
74. }
75. return n;
76. }
77. void preorder(node \*root)
78. {
79. cout<<root->data<<" ";
80. if(root->l!='\0')
81. preorder(root->l);
82. if(root->r!='\0')
83. preorder(root->r);
84. }
85. int getbal(node \*n)
86. {
87. if(n=='\0')
88. return 0;
89. else
90. return height(n->l)-height(n->r);
91. }
92. int main()
93. {
94. char ch;
95. int n;
96. do
97. {
98. cout<<"\n enter data:";
99. cin>>n;
100. root=insertion(root,n);
101. cout<<"\n continue?";
102. cin>>ch;
103. }while(ch!='n');
104. cout<<"\n preorder is:";
105. preorder(root);
106. return 0;
107. }