```
 1: #include<iostream>
 2: #include<stdlib.h>
 3: #include<queue>
 4: using namespace std;
 5:
 6: class node
 7: {
 8:     public:
 9:         node *left,*right;
10:         int data;
11: };
12: class Breadthfs
13: {
14:     public:
15:         node *insert(node*,int);
16:         void bfs(node*);
17: };
18:
19: node *insert(node *root,int data)
20: {
21:     if(!root)
22:     {
23:         root = new node;
24:         root->left=NULL;
25:         root->right=NULL;
26:         root->data=data;
27:         return root;
28:     }
29:     queue<node *> q;
30:     q.push(root);
31:     while(!q.empty())
32:     {
33:         node *temp=q.front();
34:         q.pop();
35:         if(temp->left==NULL)
36:         {
37:             temp->left=new node;
38:             temp->left->left=NULL;
39:             temp->left->right=NULL;
40:             temp->left->data=data;
41:             return root;
42:         }
43:         else{
44:             q.push(temp->left);
45:         }
46:         if(temp->right==NULL)
47:         {
48:             temp->right=new node;
49:             temp->right->left=NULL;
50:             temp->right->right=NULL;
51:             temp->right->data=data;
52:             return root;
53:         }
54:         else{
55:             q.push(temp->right);
```

```
        }
    }
}

void bfs(node *head)
{
    queue<node *>q;
    q.push(head);

    while(!q.empty())
    {

        #pragma omp parallel  for
        for(int i=0;i<q.size();i++)
        {
            node *currNode;
            #pragma omp critical
            {
                currNode=q.front();
                q.pop();
                cout<<"\t"<<currNode->data;
            }
            #pragma omp critical
            {
                if(currNode->left)
                    q.push(currNode->left);
                if(currNode->right)
                    q.push(currNode->right);
            }
        }
    }
}
int main()
{
    node *root=NULL;
    int data;
    char ans;

    do{
        cout<<"\nEnter the data: ";
        cin>>data;
        root=insert(root,data);
        cout<<"Do you want to insert one more node: ";
        cin>>ans;
    }while(ans=='Y'||ans=='y');
    bfs(root);
    return 0;
}
```