Code:

```
#include<iostream>
#include<stdlib.h>
#include<queue>
using namespace std;


class node
{
  public:

   node *left, *right;
   int data;

};

class Breadthfs
{

 public:

 node *insert(node *, int);
 void bfs(node *);

};


node *insert(node *root, int data)
// inserts a node in tree
{

   if(!root)
   {

         root=new node;
         root->left=NULL;
         root->right=NULL;
         root->data=data;
         return root;
```

```
}

queue<node *> q;
q.push(root);

while(!q.empty())
{

    node *temp=q.front();
    q.pop();

    if(temp->left==NULL)
    {

        temp->left=new node;
        temp->left->left=NULL;
        temp->left->right=NULL;
        temp->left->data=data;
        return root;
    }
    else
    {

    q.push(temp->left);

    }

    if(temp->right==NULL)
    {

        temp->right=new node;
        temp->right->left=NULL;
        temp->right->right=NULL;
        temp->right->data=data;
        return root;
    }
    else
    {

    q.push(temp->right);
```

```
        }

    }

}


void bfs(node *head)
{

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

        int qSize;

        while (!q.empty())
        {
                qSize = q.size();
                #pragma omp parallel for
                //creates parallel threads
                for (int i = 0; i < qSize; i++)
                {
                        node* currNode;
                        #pragma omp critical
                        {
                          currNode = q.front();
                          q.pop();
                          cout<<"\t"<<currNode->data;

                        }// prints parent node
                        #pragma omp critical
                        {
                        if(currNode->left)// push parent's left node in queue
                                q.push(currNode->left);
                        if(currNode->right)
                                q.push(currNode->right);
                        }// push parent's right node in queue

                }
```

```
        }

}

int main(){

   node *root=NULL;
   int data;
   char ans;

   do
   {
        cout<<"\n enter data=>";
        cin>>data;

        root=insert(root,data);

        cout<<"do you want insert one more node?";
        cin>>ans;

   }while(ans=='y'||ans=='Y');

   bfs(root);

   return 0;
}
```

Run Commands:

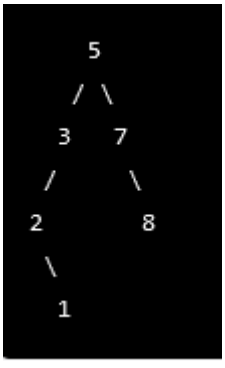
1) g++ -fopenmp bfs.cpp -o bfs

2) ./bfs

Output:

```
Enter data => 5
Do you want to insert one more node? (y/n) y

Enter data => 3
Do you want to insert one more node? (y/n) y

Enter data => 2
Do you want to insert one more node? (y/n) y

Enter data => 1
Do you want to insert one more node? (y/n) y

Enter data => 7
Do you want to insert one more node? (y/n) y

Enter data => 8
Do you want to insert one more node? (y/n) n

        5       3       7       2       1       8
```

Starting with the root node containing value 5:



The traversal would be:

```
5, 3, 7, 2, 8, 1
```