Practical 1 : BFS (Sequential and Parallel)

```
#include <iostream>
#include <queue>
#include <omp.h>
#include <chrono>
#include <cstdlib>

using namespace std;
using namespace std::chrono;

class node
{
public:
   node *left, *right;
   int data;
};

node *insert(node *root, int data)
{
   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);
      }
   }

   return root;
}

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

```
    q.push(head);

    while (!q.empty())
    {
        node *currNode = q.front();
        q.pop();
        cout<<currNode->data<<", ";

        if (currNode->left)
            q.push(currNode->left);
        if (currNode->right)
            q.push(currNode->right);
    }
}

int main()
{
    cout << "This is Atharva Pingale's Code";
    cout << "\nPractical 1 : BFS ( Sequential and Parallel )";
    node *root = NULL;
    node *root2 = NULL;
    int data;
    long int n, i;
                double start_time, end_time;
    cout << "\n\nEnter number of nodes : ";
    cin >> n;
    for (i = 0; i < n; i++)
    {
        int random_value = (rand() % (999999 - 999 + 1) + 999);
        root = insert(root, random_value);
        root2 = insert(root2, random_value);
    }

    // Sequential BFS timing
    start_time = omp_get_wtime();
    bfs(root);
    end_time = omp_get_wtime();
    double seq_time = end_time - start_time;

    // Parallel BFS timing
    start_time = omp_get_wtime();
    queue<node *> q;
    q.push(root2);

    bool empty_flag = false;

#pragma omp parallel
    {
        while (true)
        {
            node *currNode;
            bool local_empty_flag = false;
            #pragma omp critical
            {
                if (!q.empty())
                {
                    currNode = q.front();
                    q.pop();
                }
                else
                {
                    local_empty_flag = true;
                }
            }

            #pragma omp critical
            {
                empty_flag = empty_flag || local_empty_flag;
            }

            if (empty_flag)
```

```
            break;

        #pragma omp single nowait
        {
            cout << "\t" << currNode->data; // Print the node
        }

        #pragma omp critical
        {
            if (currNode->left)
                q.push(currNode->left); // Push the left child
            if (currNode->right)
                q.push(currNode->right); // Push the right child
        }
    }
  }

  end_time = omp_get_wtime();
  double parallel_time = end_time - start_time;

  cout << "\n\nSequential BFS Time: " << seq_time << " seconds";
  cout << "\n\nParallel BFS Time: " << parallel_time << " seconds\n";

  delete root;
  delete root2;

  return 0;
}
```

Output :

```
athar@LAPTOP-U0997R48 MINGW64 /d/GitHub/BE-8th-Semester/hpc_practicals (main)
$ g++ -fopenmp BFS.cpp -o BFS

athar@LAPTOP-U0997R48 MINGW64 /d/GitHub/BE-8th-Semester/hpc_practicals (main)
$ ./BFS
This is Atharva Pingale's Code
Practical 1 : BFS ( Sequential and Parallel )

Enter number of nodes : 500
1040, 19466, 7333, 27499, 20168, 16723, 12477, 30357, 27961, 25463, 6704, 29144, 24280, 17826, 10960, 1490, 3994, 12941, 5826, 6435, 33390, 15603, 4901, 1152, 1291, 13381, 18420, 19715, 20717, 20894, 6446, 22725, 1
5770, 12537, 2868, 20911, 26666, 27298, 18034, 10893, 29702, 24810, 32321, 31332, 18672, 5663, 16140, 8710, 29252, 7867, 26546, 28643, 33661, 33756, 21036, 13858, 9722, 10740, 28528, 1777, 13315, 4034, 23189, 2841,
 1287, 31105, 10039, 9941, 20263, 23647, 28445, 24804, 16889, 7728, 25369, 16349, 16005, 32100, 25392, 4547, 20628, 13622, 25083, 20953, 19755, 12839, 5965, 8375, 14930, 27307, 17943, 33438, 25625, 12322, 6536, 225
37, 17117, 3081, 23928, 17540, 5832, 32114, 5638, 30657, 23703, 10929, 14976, 3305, 32672, 23385, 6020, 29744, 27923, 20071, 7269, 6828, 27776, 16572, 6096, 17511, 24985, 14289, 10160, 19635, 23354, 25766, 24654, 1
6573, 5030, 13051, 28349, 2149, 17940, 22723, 14965, 4429, 32106, 31190, 19006, 12336, 16456, 13286, 28752, 11382, 15944, 9908, 33208, 10757, 25220, 19587, 7421, 25945, 28505, 14029, 17412, 30167, 1899, 33590, 1976
1, 2654, 18409, 7358, 28623, 21536, 22547, 7482, 28594, 5040, 4601, 25349, 11290, 31835, 10373, 12019, 5595, 25020, 28347, 24198, 20667, 25483, 9280, 5733, 1052, 2998, 27417, 28937, 7899, 4787, 19126, 1466, 4727, 1
5892, 25647, 23482, 18806, 3420, 15309, 7616, 23812, 10513, 15308, 8615, 19934, 18450, 21599, 6248, 17518, 32555, 23797, 31302, 7223, 12007, 6843, 33608, 15988, 33701, 4194, 21484, 4092, 15342, 31522, 2586, 30313,
 10502, 8447, 26199, 14457, 7617, 21579, 20795, 15797, 16280, 20588, 21797, 29008, 28156, 21471, 24621, 19537, 13291, 7037, 25178, 19189, 30656, 8957, 7190, 20814, 23887, 20155, 12510, 17201, 3633, 25271, 21054, 213
27, 23645, 27361, 5885, 19874, 29432, 30868, 21141, 24843, 2415, 22880, 32997, 11321, 19650, 11020, 6698, 4556, 29475, 28891, 25388, 6074, 11711, 3599, 3509, 22002, 27868, 18860, 15687, 14400, 10788, 16254, 17422,
 6001, 11584, 25181, 11284, 28087, 32425, 29616, 24756, 10831, 31931, 5168, 3153, 26720, 18188, 20975, 32328, 3367, 29691, 22424, 11554, 4433, 17548, 8440, 10511, 31144, 19059, 22717, 4752, 17138, 13422, 17278, 2699
5, 17686, 13528, 23548, 18436, 20865, 13948, 1192, 24194, 4296, 21415, 29285, 17104, 25487, 17281, 13454, 26733, 19113, 12700, 32315, 21670, 6785, 13262, 5312, 25354, 32184, 21052, 1911, 11807, 2831, 21944, 5312, 2
8755, 29320, 20557, 24645, 28981, 1480, 5143, 24195, 21221, 8128, 3160, 6534, 21449, 12172, 11465, 13043, 22658, 27291, 27438, 18252, 21023, 27153, 30509, 5744, 21648, 14185, 9312, 5473, 29021, 3167, 15017, 19786,
 10904, 18957, 8390, 11201, 4624, 27476, 5413, 10313, 26823, 30333, 26873, 25371, 21158, 12832, 29069, 8486, 29296, 8517, 9176, 18772, 33269, 2762, 3667, 18191, 14984, 4101, 9479, 30212, 8626, 5801, 5098, 31526, 362
4, 2542, 2923, 12022, 30971, 14060, 15180, 32002, 28431, 18504, 28592, 23724, 14030, 9491, 1141, 18221, 32285, 14063, 8899, 20186, 9359, 23412, 31973, 15269, 30169, 1234, 31832, 20710, 26759, 19895, 5666, 8284, 135
49, 1139, 14693, 3694, 22623, 29018, 3124, 27575, 22693, 23657, 27301, 18370, 23465, 5677, 23592, 24850, 26483, 2017, 29463, 22118, 24151, 3799, 19086, 32059, 2925, 10009, 5756, 33169, 21314, 10575, 31226, 13042, 2
3757, 8163, 6108, 8881, 18085, 30564, 4486, 30576, 15473, 3624, 26626, 6628, 32927, 26422, 29519, 7901, 15961, 1122, 25595, 4736, 14260, 11194, 33524,  1040

Sequential BFS Time: 0.072 seconds

Parallel BFS Time: 0 seconds

athar@LAPTOP-U0997R48 MINGW64 /d/GitHub/BE-8th-Semester/hpc_practicals (main)
$
```

Sequential BFS Time : 0.072 seconds
Parallel BFS Time : 0 seconds