Assignment – 1

```
Code:
#include <bits/stdc++.h>
#include <omp.h>
#include <chrono>
using namespace std;
using namespace std::chrono;
// Node structure for the binary tree
struct Node {
  int val;
  Node* left;
  Node* right;
  Node(int v) {
    val = v;
    left = right = NULL;
  }
};
// Sequential Depth First Search (DFS) traversal of the binary tree
void sequentialDFS(Node* root) {
  if (!root) return;
  stack<Node*>s;
  s.push(root);
  while (!s.empty()) {
    Node* curr = s.top();
    s.pop();
    cout << curr->val << " ";
    if (curr->right)
      s.push(curr->right);
    if (curr->left)
      s.push(curr->left);
  }
  cout << endl;
}
// Sequential Breadth First Search (BFS) traversal of the binary tree
```

```
void sequentialBFS(Node* root) {
  if (!root) return;
  queue<Node*> q;
  q.push(root);
  while (!q.empty()) {
    Node* curr = q.front();
    q.pop();
    cout << curr->val << " ";
    if (curr->left)
       q.push(curr->left);
    if (curr->right)
      q.push(curr->right);
  }
  cout << endl;
// Parallel Depth First Search (DFS) traversal of the binary tree
void parallelDFS(Node* root) {
  if (!root) return;
  stack<Node*>s;
  s.push(root);
  #pragma omp parallel
    while (!s.empty()) {
       Node* curr;
      #pragma omp critical
         curr = s.top();
         s.pop();
       }
      cout << curr->val << " ";
       #pragma omp single
         if (curr->right)
           s.push(curr->right);
         if (curr->left)
```

```
s.push(curr->left);
      }
    }
  }
  cout << endl;
}
// Parallel Breadth First Search (BFS) traversal of the binary tree
void parallelBFS(Node* root) {
  if (!root) return;
  queue<Node*> q;
  q.push(root);
  #pragma omp parallel
    while (!q.empty()) {
       Node* curr;
      #pragma omp critical
         curr = q.front();
         q.pop();
      cout << curr->val << " ";
      #pragma omp single
        if (curr->left)
           q.push(curr->left);
        if (curr->right)
           q.push(curr->right);
      }
    }
  }
  cout << endl;
}
// Function to measure the time taken by a traversal function
double measureTime(void (*function)(Node*), Node* root) {
  auto start = chrono::high_resolution_clock::now();
  function(root);
```

```
auto end = chrono::high resolution clock::now();
  chrono::duration<double> duration = end - start;
  return duration.count();
}
signed main() {
  // Prompting the user to enter the value for the root node
  int rootVal;
  cout << "Enter the value for the root node: ";
  cin >> rootVal;
  Node* root = new Node(rootVal);
  // Queue to store nodes whose children need to be entered
  queue<Node*> pendingNodes;
  pendingNodes.push(root);
  // Prompting the user to enter values for each node and constructing the binary tree
  while (!pendingNodes.empty()) {
    Node* curr = pendingNodes.front();
    pendingNodes.pop();
    int leftVal, rightVal;
    cout << "Enter the value for the left child of " << curr->val << " (-1 if none): ";
    cin >> leftVal;
    if (leftVal != -1) {
      curr->left = new Node(leftVal);
      pendingNodes.push(curr->left);
    }
    cout << "Enter the value for the right child of " << curr->val << " (-1 if none): ";
    cin >> rightVal;
    if (rightVal != -1) {
      curr->right = new Node(rightVal);
      pendingNodes.push(curr->right);
    }
  }
  // Menu-driven program to allow the user to choose traversal methods
  bool flag = true;
  while(flag) {
    cout << "\n<----->" << endl;
```

```
cout << "2. For Sequential BFS and Parallel BFS" << endl;
    cout << "3. For Exit" << endl;
    int ch;
    cout << "Enter Your Choice: ";
    cin >> ch;
    switch(ch) {
       case 1: {
         cout << "\nSequential DFS : ";</pre>
         double timeSequentialDFS = measureTime(sequentialDFS, root);
         \verb|cout| << \verb|Time| taken for Sequential DFS: " << \verb|timeSequential DFS| << " seconds" << \verb|end|; \\
         cout << "\nParallel DFS : ";</pre>
         double timeParallelDFS = measureTime(parallelDFS, root);
         cout << "Time taken for Parallel DFS: " << timeParallelDFS << " seconds" << endl;
         if(timeParallelDFS < timeSequentialDFS) cout << "\nParallel DFS performing better than
Sequential DFS." << endl;
         else cout << "\nSequential DFS is performing better than Parallel DFS" << endl;
         break;
       }
       case 2: {
         cout << "\nSequential BFS : ";
         double timeSequentialBFS = measureTime(sequentialBFS, root);
         cout << "Time taken for Sequential BFS: " << timeSequentialBFS << " seconds" << endl;</pre>
         cout << "Parallel BFS : ";</pre>
         double timeParallelBFS = measureTime(parallelBFS, root);
         cout << "Time taken for Parallel BFS: " << timeParallelBFS << " seconds" << endl;
         if(timeParallelBFS < timeSequentialBFS) cout << "\nParallel BFS performing better than
Sequential BFS." << endl;
         else cout << "\nSequential BFS is performing better than Parallel BFS" << endl;
         break:
         break;
       case 3: {
         cout << "\nThank You!!" << endl;</pre>
         flag = false;
         break;
```

cout << "1. For Sequential DFS and Parallel DFS" << endl;

```
}
    default: {
        cout << "\nInvalid choice, Try again" << endl;
        break;
    }
    }
}
return 0;
}
</pre>
```

Output:

```
Windows PowerShell
PS E:\AIT\4th Year Sem 2> g++ 7446_Assignment_1.cpp -lgomp -o as
PS E:\AIT\4th Year Sem 2> ./as
Enter the value for the root node: 5
Enter the value for the left child of 5 (-1 if none): 6
Enter the value for the right child of 5 (-1 if none): 8
Enter the value for the left child of 6 (-1 if none): 18
Enter the value for the right child of 6 (-1 if none): 9 Enter the value for the left child of 8 (-1 if none): 10
Enter the value for the right child of 8 (-1 if none): 12 Enter the value for the left child of 18 (-1 if none): -1
Enter the value for the right child of 18 (-1 if none): -1
Enter the value for the left child of 9 (-1 if none): -1
Enter the value for the right child of 9 (-1 if none): -1
Enter the value for the left child of 10 (-1 if none): -1
Enter the value for the right child of 10 (-1 if none): -1
Enter the value for the left child of 12 (-1 if none): -1
Enter the value for the right child of 12 (-1 if none): -1
<----MENU-

    For Sequential DFS and Parallel DFS
    For Sequential BFS and Parallel BFS

3. For Exit
Enter Your Choice: 1
Sequential DFS : 5 6 18 9 8 10 12
Time taken for Sequential DFS: 0.001139 seconds
Parallel DFS : 5 6 18 9 8 10 12
Time taken for Parallel DFS: 0 seconds
Parallel DFS performing better than Sequential DFS.
       --MENU---
1. For Sequential DFS and Parallel DFS
2. For Sequential BFS and Parallel BFS
3. For Exit
Enter Your Choice: 2
Sequential BFS : 5 6 8 18 9 10 12
Time taken for Sequential BFS: 0.002074 seconds
Parallel BFS : 5 6 8 18 9 10 12
Time taken for Parallel BFS: 0 seconds
Parallel BFS performing better than Sequential BFS.
<----MENU-
1. For Sequential DFS and Parallel DFS
2. For Sequential BFS and Parallel BFS
3. For Exit
Enter Your Choice: 3
Thank You!!
PS E:\AIT\4th Year Sem 2>
```