**The laboratory work 7**

**Depth-first search** always expands the *deepest* node in the current frontier of the search tree. The search proceeds immediately to the deepest level of the search tree, where the nodes have no successors. As those nodes are expanded, they are dropped from the frontier, so then the search “backs up” to the next deepest node that still has unexplored successors.

Whereas breadth-first-search uses a FIFO queue, depth-first search uses a LIFO queue.

A LIFO queue means that the most recently generated node is chosen for expansion. This

must be the deepest unexpanded node because it is one deeper than its parent—which, in turn, was the deepest unexpanded node when it was selected.



Assignment – write a program code that will realize the traversal of the binary search tree. In the first part, we need to create a class of the binary tree that will keep data about the structure of the binary tree.

using namespace std;

class Node {

 int data;

 Node \*left;

 Node \*right;

public:

 Node() { data = -1; left = NULL; right = NULL; };

 void setKey(int key) { data = key; };

 void setLeft(Node\* Left) { left = Left; };

 void setRight(Node\* Right) { right = Right; };

 int getKey() { return data; }

 Node \*getLeft() { return left; }

 Node \*getRight() { return right; }

}

Then you will need to create addNode function to insert a new node in the binary tree and levelOrder to traverse the binary tree as the breadth first search.

int main()

{

 Node \*tree = new Node();

 tree->addNode(5, tree);

 tree->addNode(4, tree);

 tree->addNode(36, tree);

 tree->addNode(3, tree);

 tree->addNode(34, tree);

 tree->addNode(8, tree);

 tree->addNode(1, tree);

 cout << "The binary tree we have is " << endl;

 tree->levelOrder(tree);

 return 0;

}