Nama kelompok:

- Muhammad Zidan Alif Oktavian (045)
- Dika Wahyu Nuralixsyah (029)
- Fahreza Dzaky Rahmatullah (061)
- Ariel Fikri Ramadhani (067)
- Bariq Adyatma (019)

Coding:

```
#include<iostream>
#define SPACE 10

using namespace std;
//mendeklarasikan class
class TreeNode {
  public:
    int value;
  TreeNode * left;
  TreeNode * right;

  TreeNode() {
    value = 0;
    left = NULL;
    right = NULL;
  }
  TreeNode(int v) {
    value = v;
    left = NULL;
    right = NULL;
  }
};

class BST {
  public:
    TreeNode * root;
  BST() {
    root = NULL;
  }
  bool isTreeEmpty() {
    if (root == NULL) {
      return true;
    } else {
      return false;
    }
  }
```

```
// fungsi untuk menambahkan nilai ke tree
  void insertNode(TreeNode * new_node) {
    if (root == NULL) {
      root = new_node;
      cout << "Value Inserted as root node!" << endl;
    } else {
      TreeNode * temp = root;
      while (temp != NULL) {
        if (new_node -> value == temp -> value) {
          cout << "Nilai telah ada," <<
            "masukkan nilai yang lain" << endl;
          return;
        } else if ((new_node -> value < temp -> value) && (temp -> left == NULL)) {
          temp -> left = new_node;
          cout << "nilai dimasukkan ke kiri" << endl;
          break;
        } else if (new_node -> value < temp -> value) {
          temp = temp -> left;
        } else if ((new_node -> value > temp -> value) && (temp -> right == NULL)) {
          temp -> right = new_node;
          cout << "nilai dimasukkan ke kanan" << endl;
          break;
        } else {
          temp = temp -> right;
        }
      }
    }
  }
  TreeNode* insertRecursive(TreeNode *r, TreeNode *new_node)
  {
      if(r==NULL)
      {
          r=new_node;
          cout <<"sukses memasukkan nilai"<<endl;
          return r;
      }
```

```
    if(new_node->value < r->value)
    {
        r->left = insertRecursive(r->left,new_node);
    }
    else if (new_node->value > r->value)
    {
        r->right = insertRecursive(r->right,new_node);
    }
   else
    {
        cout << "tidak boleh ada nilai yang sama" << endl;
        return r;
    }
    return r;
  }

void print2D(TreeNode * r, int space) {
  if (r == NULL) // Base case
    return;
  space += SPACE; // meningkatkan jarak diantara angka
  print2D(r -> right, space); // memproses children atau subtree disebelah kanan terlebih dahulu
  cout << endl;
  for (int i = SPACE; i < space; i++)
    cout << " ";
  cout << r -> value << "\n";
  print2D(r -> left, space); // memproses children disebelah kiri
}

void printPreorder(TreeNode * r) //mengurutkan dari(nilai sekarang, kiri, kanan)
{
  if (r == NULL)
    return;
  /* print nilai pertama terlebih dahulu*/
  cout << r -> value << " ";
  /* rekursif dari kiri*/
  printPreorder(r -> left);
  /* lalu dilanjutkan rekursif dari kanan */
  printPreorder(r -> right);
}

void printInorder(TreeNode * r) //mengurutkan dari  (kiri, nilai sekarang, kanan)
{
  if (r == NULL)
    return;
  /* rekursif dari kiri */
  printInorder(r -> left);
  /* print data yang sekarang*/
  cout << r -> value << " ";
  /* rekursif dari kanan*/
  printInorder(r -> right);
}
void printPostorder(TreeNode * r) //mengurutkan dari (kiri, kanan, Root)
{
  if (r == NULL)
    return;
  // rekursif dari kiri
  printPostorder(r -> left);
  // rekursif dari kanan
  printPostorder(r -> right);
  // print data yang sekarang
  cout << r -> value << " ";
}

TreeNode * iterativeSearch(int v) {
  if (root == NULL) {
    return root;
  } else {
    TreeNode * temp = root;
    while (temp != NULL) {
      if (v == temp -> value) {
        return temp;
      } else if (v < temp -> value) {
        temp = temp -> left;
      } else {
        temp = temp -> right;
      }
    }
    return NULL;
  }
}

TreeNode * recursiveSearch(TreeNode * r, int val) {
  if (r == NULL || r -> value == val)
    return r;

  else if (val < r -> value)
    return recursiveSearch(r -> left, val);

  else
    return recursiveSearch(r -> right, val);
}

int height(TreeNode * r) {
  if (r == NULL)
    return -1;
  else {
    /* menghitung tinggi setiap subtree */
    int lheight = height(r -> left);
    int rheight = height(r -> right);

    /* gunakan yang terbesar */
    if (lheight > rheight)
      return (lheight + 1);
    else return (rheight + 1);
  }
}
```

```
  /* Print angka/nilai pada tempat yang telah ditentukan*/
  void printGivenLevel(TreeNode * r, int level) {
    if (r == NULL)
      return;
    else if (level == 0)
      cout << r -> value << " ";
    else  level > 0;
    {
      printGivenLevel(r -> left, level - 1);
      printGivenLevel(r -> right, level - 1);
    }
  }
  void printLevelOrderBFS(TreeNode * r) {
    int h = height(r);
    for (int i = 0; i <= h; i++)
      printGivenLevel(r, i);
  }

  TreeNode * minValueNode(TreeNode * node) {
    TreeNode * current = node;
    /* loop down to find the leftmost leaf */
    while (current -> left != NULL) {
      current = current -> left;
    }
    return current;
  }



};
int main() {
  BST obj;
  int option, val;

  do {
    cout << "What operation do you want to perform? " <<
      " Select Option number. Enter 0 to exit." << endl;
    cout << "1. Insert Node" << endl;
    cout << "2. Print/Traversal BST values" << endl;
    cout << "0. Exit Program" << endl;

    cin >> option;
    TreeNode n1;
    TreeNode * new_node = new TreeNode();

    switch (option) {
    case 0:
      break;
    case 1:
        cout <<"INSERT"<<endl;
          cout <<"Masukkan nilai satu persatu: ";
          cin >> val;
          new_node->value = val;
          obj.root= obj.insertRecursive(obj.root,new_node);
          obj.insertNode(new_node);
          cout<<endl;
            break;


    case 2:
      cout << "PRINT 2D: " << endl;
      obj.print2D(obj.root, 5);
      cout << endl;
      cout << "Print Level Order BFS: \n";
      obj.printLevelOrderBFS(obj.root);
      cout << endl;
      cout <<"PRE-ORDER: ";
      obj.printPreorder(obj.root);
      cout<<endl;
      cout <<"IN-ORDER: ";
      obj.printInorder(obj.root);
      cout<<endl;
      cout <<"POST-ORDER: ";
      obj.printPostorder(obj.root);
      break;

    }

  } while (option != 0);

  return 0;
}
```

input

```
INSERT
Masukkan nilai satu persatu: 1
sukses memasukkan nilai
Nilai telah ada,masukkan nilai yang lain
```

```
What operation do you want to perform?  Select Option number. Enter 0 to exit.
1. Insert Node
2. Print/Traversal BST values
0. Exit Program
1
INSERT
Masukkan nilai satu persatu: 2
sukses memasukkan nilai
Nilai telah ada,masukkan nilai yang lain
```

```
What operation do you want to perform?  Select Option number. Enter 0 to exit.
1. Insert Node
2. Print/Traversal BST values
0. Exit Program
1
INSERT
Masukkan nilai satu persatu: 6
sukses memasukkan nilai
Nilai telah ada,masukkan nilai yang lain
```

output

```
What operation do you want to perform?  Select Option number. Enter 0 to exit.
1. Insert Node
2. Print/Traversal BST values
0. Exit Program
2
PRINT 2D:

                        6

              2

    1

Print Level Order BFS:
1 2 6
PRE-ORDER: 1 2 6
IN-ORDER: 1 2 6
POST-ORDER: 6 2 1 What operation do you want to perform?  Select Option number. Enter 0 to exit.
```