POST TEST PRAKTIKUM

STRUKTUR DATA DAN ALGORITMA

Disusun Oleh:

Riyon Aryono **: 2211102241**

Dosen

Muhamad Azrino Gustalika

PROGRAM STUDI S1 TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM
PURWOKERTO

2023

1. Dibawah ada beberapa string yang akan disimpan ke dalam hash-table, jumlah hashtable ada 30 index, cari index hash-table dimana string akan disimpan dengan menggunakan division method :
   a. AJOJING
   b. CHARLIE

   Source Code

```
#include <iostream>
#include <string>
using namespace std;

// ini Fungsi dari Division Method
int division(const string &key, int tableSize)
{
    int sum = 0;
    for (char c : key)
    {
        sum += c;
    }
    return sum % tableSize;
}

int main()
{
    string ajojing = "AJOJING";
    string charlie = "CHARLIE";
    int TABLE_SIZE = 30;

    int indexAjojing = division(ajojing, TABLE_SIZE);
    int indexCharlie = division(charlie, TABLE_SIZE);

    cout << ajojing << " ditemukan di index: " << indexAjojing<<  endl;
    cout << charlie << " ditemukan di index: " << indexCharlie<<  endl;

    return 0;
}
```

Output

```
AJOJING ditemukan di index: 4
CHARLIE ditemukan di index: 24
PS C:\Users\yggdrasil\Documents\Me\Kuliah\SMSTR2\prak algodat\
```

2. Carilah urutan In-order dan Postorder dari Binary Tree di bawah ini :

Source Code

```
#include <iostream>

using namespace std;

// Program Binary Tree

// Deklarasi Pohon
struct Pohon
{
   char data;
   Pohon *left, *right, *parent;
};

Pohon *root, *baru;

// Inisialisasi
void init()
{
   root = NULL;
}

// Cek Node
int isEmpty()
{
   if (root == NULL)
   {
      return 1;
   }
   else
   {
      return 0;
   }
}

// Buat Node Baru
void buatNode(char data)
{
   if (isEmpty() == 1)
   {
      root = new Pohon();
      root->data = data;
      root->left = NULL;
      root->right = NULL;
```

```
        root->parent = NULL;
    }
    else
    {
        cout << "\nPohon sudah dibuat" << endl;
    }
}

// Tambah Kiri
Pohon *insertLeft(char data, Pohon *node)
{
    if (isEmpty() == 1)
    {
        cout << "\nBuat tree terlebih dahulu!" << endl;
        return NULL;
    }
    else
    {
        // cek apakah child kiri ada atau tidak
        if (node->left != NULL)
        {
            // kalau ada
            cout << "\nNode " << node->data << " sudah ada di child kiri!" <<
   endl;
        }
        else
        {
            // Kalau tidak ada
            baru = new Pohon();
            baru->data = data;
            baru->left = NULL;
            baru->right = NULL;
            baru->parent = node;
            node->left = baru;
            return baru;
        }
    }
}

// Tambah kanan
Pohon *insertRight(char data, Pohon *node)
{
    if (root == NULL)
    {
        cout << "\nBuat tree terlebih dahulu!" << endl;
        return NULL;
```

```
    }
    else
    {
        // cek apakah child kanan ada atau tidak
        if (node->right != NULL)
        {
            // kalau ada
            cout << "\nNode " << node->data << " sudah ada di child kanan!"
   << endl;
        }
        else
        {
            // Kalau tidak ada
            baru = new Pohon();
            baru->data = data;
            baru->left = NULL;
            baru->right = NULL;
            baru->parent = node;
            node->right = baru;
            return baru;
        }
    }
}


// in Order
void inOrder(Pohon *node = root)
{
    if (!root)
    {
        cout << "\n Buat tree terlebih dahulu" << endl;
    }
    else
    {
        if (node != NULL)
        {
            inOrder(node->left);
            cout << " " << node->data << ", ";
            inOrder(node->right);
        }
    }
}

// Post Order
void postOrder(Pohon *node = root)
{
```

```
    if (!root)
    {
        cout << "\n Buat tree terlebih dahulu" << endl;
    }
    else
    {
        if (node != NULL)
        {
            postOrder(node->left);
            postOrder(node->right);
            cout << " " << node->data << ", ";
        }
    }
}


int main()
{
    buatNode('A');

    Pohon *nodeB, *nodeC, *nodeD, *nodeE, *nodeF, *nodeG, *nodeH,
   *nodeZ;

    nodeB = insertLeft('B', root);
    nodeF = insertRight('F', root);
    nodeC = insertLeft('C', nodeB);
    nodeE = insertRight('E', nodeF);
    nodeH = insertRight('H', nodeC);
    nodeD = insertLeft('D', nodeE);
    nodeG = insertRight('G', nodeE);
    nodeZ = insertLeft('Z', nodeH);

    cout << "\n InOrder :" << endl;
    inOrder(root);

    cout << "\n PostOrder :" << endl;
    postOrder(root);
}
```

Output

```
 InOrder :
 C,  Z,  H,  B,  A,  F,  D,  E,  G,
 PostOrder :
 Z,  H,  C,  B,  D,  G,  E,  F,  A,
PS C:\Users\yggdrasil\Documents\Me\Kuliah\SMSTR2\prak algodat\
```

3. Perbaiki Program yang sudah ada agara mengahasilkan output

   3258, 2515, 6123

   Source Code

```
#include <iostream>
using namespace std;

int main()
{
   int riyon_2211102241;
   int arr[3][4] = {{3, 2, 5, 8}, {2, 5, 1, 5}, {6, 1, 2, 3}};
   for (int i = 0; i < 3; i++)
   {
      for (int j = 0; j < 4; j++)
      {
         cout << arr[i][j];
      }
      cout << endl;
   }
   return 0;
}
```

Output