Tugas Pendahuluan Modul 11

STRUKTUR DATA – Ganjil 2025/2026

Tree

Ketentuan Tugas Pendahuluan

1. Tugas Pendahuluan dikerjakan secara Individu.
2. TP ini bersifat WAJIB, tidak mengerjakan = PENGURANGAN POIN JURNAL / TES ASESMEN.
3. Hanya MENGUMPULKAN tetapi TIDAK MENGERJAKAN = PENGURANGAN POIN JURNAL / TES ASESMEN.
4. Deadline pengumpulan TP Modul 11 adalah Senin, 24 November 2025 pukul 06.00 WIB.
5. Tidak ada TOLERANSI KETERLAMBATAN. TERLAMBAT atau TIDAK MENGUMPULKAN TP maka DIANGGAP TIDAK MENGERJAKAN.
6. DILARANG PLAGIAT (PLAGIAT = E).
7. Kerjakan TP dengan jelas agar dapat dimengerti.
8. Untuk setiap soal nama fungsi atau prosedur WAJIB menyertakan NIM, contoh:

```
int namaFungsi_10301XXXXXXXX(...);
```

9. File diupload di LMS menggunakan format PDF dengan format:

```
TP_MOD_[XX]_NIM_NAMA.pdf
```

CP (WhatsApp):

- Falih (+62 857-2774-5199)
- Faried (+62 813-8921-4045)
- Arief (+62 852-1252-8394)
- Evan (+62 896-0298-0999)

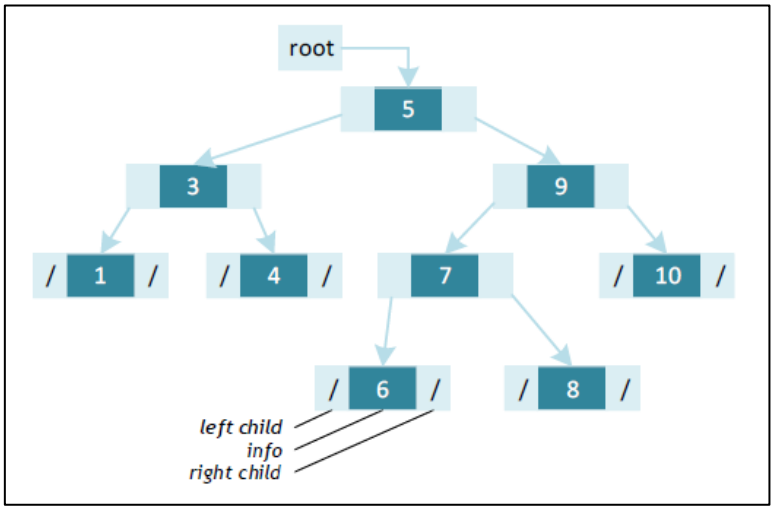
SELAMAT MENGERJAKAN!

1 Binary Search Tree

Binary Tree adalah salah satu jenis struktur tree di mana setiap node hanya dapat memiliki maksimal dua anak. Kedua anak tersebut biasanya disebut sebagai **left child** dan **right child**. Struktur binary tree sering diolah menggunakan **algoritma rekursif** karena bentuknya yang hierarkis dan berulang.

Binary Search Tree (BST) adalah bentuk khusus dari binary tree yang memiliki aturan penyusunan nilai:

- Nilai pada sebuah node lebih besar daripada seluruh nilai yang berada pada subtree kiri.
- Nilai pada sebuah node lebih kecil daripada seluruh nilai yang berada pada subtree kanan.



Buatlah program untuk membuat tree sederhana menggunakan **Binary Search Tree**!

1.1 ADT Program

Buatlah ADT Stack (`tree.h`) untuk tree nya.

```
type infotype : integer
type adrNode : pointer to Node

type Node <
    info : infotype
    left : adrNode
    right : adrNode
>
```

1.2 Fungsi/Prosedur Primitif

Buatlah fungsi/prosedur primitif (`tree.cpp`) untuk tree nya.

```
procedure createTree(in/out root : adrNode)
{I.S. -
 F.S. Terdefinisi root sebagai BST dengan nilai nil (nullptr).}
algorithm
    root <- nil
endprocedure

function createNode(x : infotype) -> adrNode
{I.S. Terdefinisi infotype x.
 F.S. Mengembalikan alamat dari suatu node hasil alokasi, dengan info
adalah x dan left dan right adalah nil (nullptr).}
dictionary
    p : adrNode
algorithm
    aloc(p)
    p.info <- x
    p.left <- nil
    p.right <- nil

    return p
endfunction


procedure insertNode(in/out root : adrNode, in adrNode p)
{I.S. Terdefinisi root dari BST, dan alamat p.
 F.S. Elemen yang ditunjuk oleh p ditambahkan sebagai node dari BST.}
algorithm
    if root == nil then
        root <- p
    else if p.info < root.info then
        insertNode(root.left, p)
    else
        insertNode(root.right, p)
    endif
endprocedure

function searchNode(root : adrNode, x : infotype) -> adrNode
{I.S. Terdefinisi root dari BST, dan infotype x.
 F.S. Mengembalikan alamat dari node yang memiliki info sama dengan x,
dan mengembalikan nil apabila tidak ditemukan.}
algorithm
    if root == nil then
        return nil
    else if x < root.info then
        return searchNode(root.left, x)
    else if x > root.info then
        return searchNode(root.right, x)
    endif

    return root
endfunction

procedure displayTree(root : adrNode)
{I.S. Terdefinisi root dari BST.
```

```
 F.S. Menampilkan isi node dalam BST dengan metode inorder traversal.}
algorithm
    if root != nil then
        displayTree(root.left)
        print(root.info, " ")
        displayTree(root.right)
    endif
endprocedure
```

1.3 Implementasi Subprogram

Buatlah subprogram (`tree.cpp`) untuk tree nya.

```
function countNodes(root : adrNode) -> integer
{I.S. Terdefinisi root dari BST.
 F.S. Mengembalikan banyak node dalam BST.}

function getMinValue(root : adrNode) -> infotype
{I.S. Terdefinisi root dari BST.
 F.S. Mengembalikan nilai node terkecil dalam BST.}

function getMaxValue(root : adrNode) -> infotype
{I.S. Terdefinisi root dari BST.
 F.S. Mengembalikan nilai terbesar dalam BST.}
```

1.4 Main Program

Implementasikan program utama seusai dengan teks dibawah (teks bold bergaris bawah adalah input).

```
Masukkan node: 50
Masukkan node: 30
Masukkan node: 70
Masukkan node: 20
Masukkan node: 40
Masukkan node: 60
Masukkan node: 80

Masukkan nilai dari node yang ingin dicari: 60
Node dengan nilai 60 ditemukan!

Print BST (inorder traversal): 20 30 40 50 60 70 80
Jumlah node: 7
Nilai terkecil: 20
Nilai terbesar: 80
```