

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_10301XXXXXXX(...);
```

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

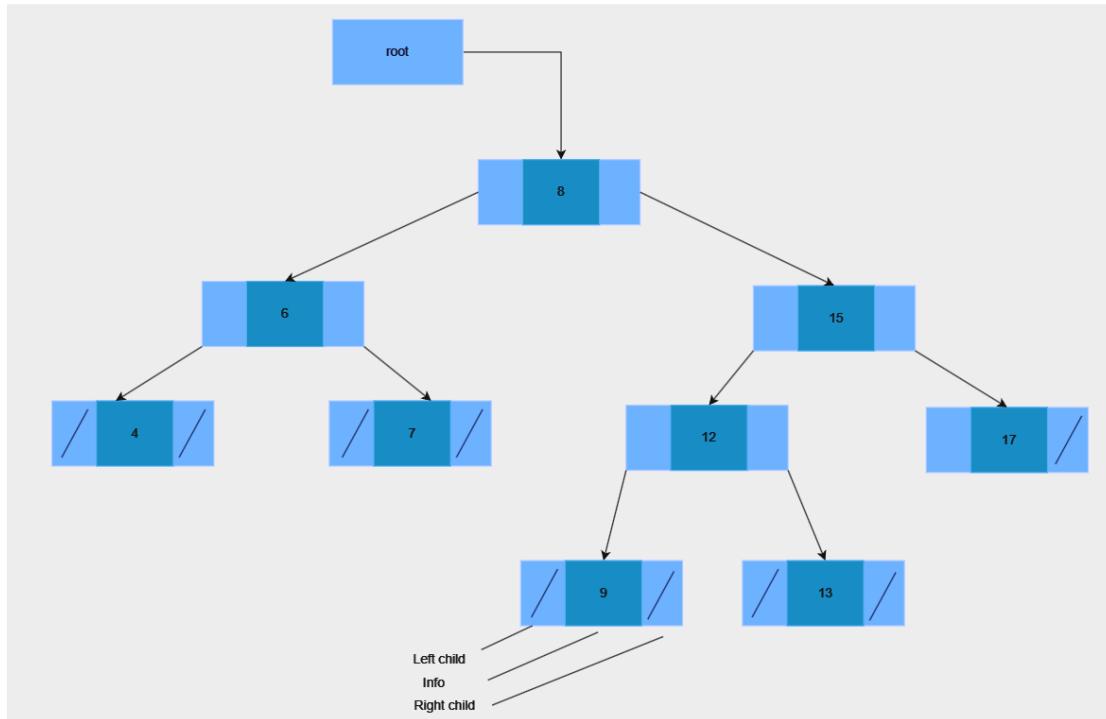
```
TP_MOD_[XX]_NIM_NAMA.pdf
```

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 Node Deletion



Buatlah program untuk melakukan pencarian node dalam **Binary Search Tree**!

1.1 ADT Program

Buatlah ADT (`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

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)
        output(root.info, " ")
        displayTree(root.right)
    endif
endprocedure

```

1.3 Implementasi Subprogram

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

```

function getMinNode(root : adrNode) -> adrNode
{I.S. Terdefinisi root dari BST.
F.S. Mengembalikan node dari BST dengan nilai terkecil.}
dictionary
    p : adrNode
algorithm
    p <- root

    while p != nil and p.left != nil do
        p <- p->left
    endwhile

```

```
    return p
endfunction

function getMaxNode(root : adrNode) -> adrNode
{I.S. Terdefinisi root dari BST.
F.S. Mengembalikan node dari BST dengan nilai terbesar.}
dictionary
    p : adrNode
algorithm
    p <- root

    while p != nil and p.left != nil do
        p <- p->left
    endwhile

    return p
endfunction
```

1.4 Main Program

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

```
Masukkan node: 8
Masukkan node: 6
Masukkan node: 15
Masukkan node: 4
Masukkan node: 7
Masukkan node: 12
Masukkan node: 17
Masukkan node: 9
Masukkan node: 13

Node dalam BST: 4 6 7 8 9 12 13 15 17
Node terkecil dalam BST: 4
Node terbesar dalam BST: 17
```