Ketentuan Tugas Pendahuluan

- Untuk soal teori **JAWABAN DIKETIK DENGAN RAPI** dan untuk soal algoritma **SERTAKAN SCREENSHOOT CODINGAN DAN HASIL OUTPUT**.
- TP ini bersifat **WAJIB**, tidak mengerjakan = **PENGURANGAN POIN JURNAL**.
- Hanya **MENGUMPULKAN** tetapi **TIDAK MENGERJAKAN = PENGURANGAN POIN**.
- Deadline pengumpulan TP Modul 15 adalah **Senin, 2 Januari 2023** pukul 06.00 WIB.
- **TIDAK ADA TOLERANSI KETERLAMBATAN, TERLAMBAT ATAU TIDAK MENGUMPULKAN TP ONLINE MAKA DIANGGAP TIDAK MENGERJAKAN**.
- **DILARANG PLAGIAT (PLAGIAT = E)**.
- Kerjakan TP dengan jelas agar dapat dimengerti.
- Untuk setiap soal nama fungsi atau prosedur **WAJIB** menyertakan **NIM**, contoh: **insertFirst_1301201111**.
- File diupload di LMS menggunakan format **PDF** dengan ketentuan **:** **TP_MODX_NIM_KELAS.pdf**

```
Contoh:
    int searchNode_130190XXXX (List L, int X);
```

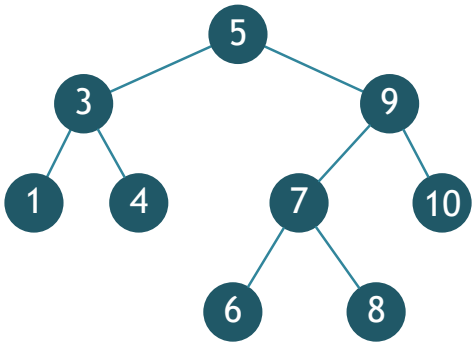
CP (WA/LINE):

- Fatan (0895610337706 / fatanaqilla144)
- Fadli Zuhri (089628906368 / fadli_zuhri)
- Moh Naufal Mizan Saputro (089505169183 / mznsptr)
- Ihsani Hawa Arsytania (082141338004 /hawa.ihsani)

SELAMAT MENGERJAKAN^^

Tugas Pendahuluan Praktikum Modul 15
Binary Search Tree

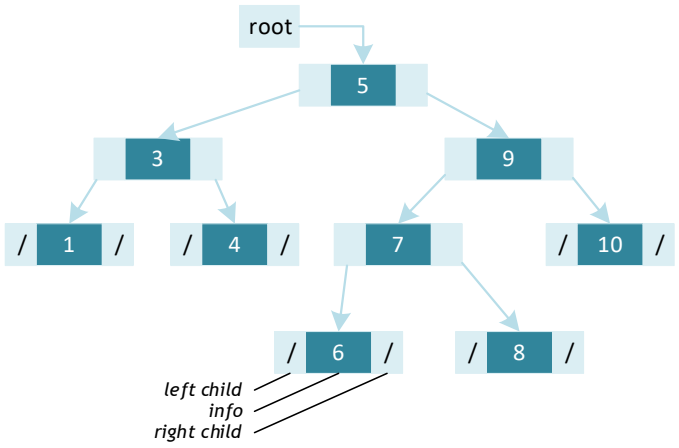
Binary Tree merupakan salah satu jenis *tree* di mana sebuah *node* pada *tree* tersebut hanya bisa memiliki anak maksimal sebanyak dua. Secara umum menggunakan istilah *left child* dan *right child* untuk penyebutan *node* anak. Pada praktikum modul ini kita akan fokus ke salah satu bentuk *binary tree*, yaitu *Binary Search Tree*. Secara umum algoritma yang digunakan menggunakan konsep rekursif, jadi silahkan pelajari kembali materi terkait rekursif yang sudah dipelajari pada mata kuliah Algoritma Pemrograman atau dari referensi lainnya!



Gambar 1. Contoh Binary Search Tree

Binary Search Tree (BST) atau disebut juga *ordered* atau *sorted binary tree* memiliki nilai pada *internal node* lebih besar dibandingkan dengan nilai *node/subtree* di sebelah kiri, dan memiliki nilai *node* yang lebih kecil dibandingkan dengan nilai *node/subtree* di sebelah kanannya (perhatikan contoh pada Gambar 1).

Representasi BST ataupun *binary tree* bisa menggunakan konsep *list*, di mana elemen *tree* terdiri dari *info* (berisi nilai dari *node*) dan dua *pointer* (*left* dan *right*) yang digunakan untuk menyimpan alamat memori dari elemen *tree* anak kiri dan kanan. Perhatikan ilustrasi berikut ini!



Gambar 2. Ilustrasi *list* dari BST pada Gambar 1

Soal Tugas Pendahuluan (Nilai Max : 10):

1. Buatlah ADT (Tree.h, dan Tree.cpp) dari BST dengan info berupa bilangan bulat seperti ilustrasi yang diberikan pada Gambar 2. (1 Poin)

2. Lengkapi primitif dari BST berikut ini pada file Tree.cpp! (8 Poin)

```
adrNode newNode(infotype x); ( 1 Poin )
/* mengembalikan alamat dari suatu node hasil alokasi, dengan info adalah x
dan left dan right adalah NULL */

adrNode findNode(adrNode root, infotype x); ( 1 Poin )
/* mengembalikan alamat dari node yang memiliki info sama dengan x, atau NULL
apabila tidak ditemukan, Catatan: implementasikan secara REKURSIF */

void insertNode(adrNode &root, adrNode p); ( 1 Poin )
/* I.S. terdefinisi root dari BST (mungkin NULL), dan pointer p yang berisi
alamat node yang mau ditambahkan pada BST
   F.S. elemen yang ditunjuk oleh p ditambahkan sebagai node dari BST
Catatan: implementasikan secara REKURSIF */

void printPreOrder(adrNode root); ( 1 Poin )
/* I.S. terdefinisi root dari BST (mungkin NULL)
   F.S. menampilkan node dari BST secara Preorder atau dengan urutan root, left
dan right
Catatan: implementasikan secara REKURSIF */

void printDescendant(adrNode root, infotype x); ( 1 Poin )
/* I.S. terdefinisi suatu root dari BST (mungkin NULL), dan infotype x
   F.S. menampilkan subtree atau keturunan dari node yang memiliki info x
Catatan: implementasikan secara REKURSIF */

int sumNode(adrNode root); ( 1 Poin )
/* mengembalikan hasil penjumlahan semua info node yang terdapat pada BST
Catatan: implementasikan secara REKURSIF */

int countLeaves(adrNode root); ( 1 Poin )
/* mengembalikan banyaknya daun atau node yang tidak memiliki anak dari BST
Catatan: implementasikan secara REKURSIF */

int heightTree(adrNode root); ( 1 Poin )
/* mengembalikan banyaknya edge dari suatu root menuju daun yang terjauh.
Catatan: implementasikan secara REKURSIF */
```

3. Buatlah *main program* main.cpp berikut ini untuk menguji *subprogram* yang telah dibuat. (1 Poin)

```
int main(){
    int x[9] = {5,3,9,10,4,7,1,8,6};
    /* Tampilkan isi dari array */
    ...
    /* 1. Tambahkan setiap elemen array x kedalam BST secara berurutan */
    /* sehingga dihasilkan BST seperti Gambar 1, gunakan looping*/
    ...
    ...
    /* 2. Tampilkan node dari BST secara Pre-Order */
    printf("\n");
    printf("\nPre Order\t: ");
    ...
    /* 3. Tampilkan keturunan dari node 9*/
    printf("\n");
    printf("\nDescendent of Node 9\t: ");
    ...
    /* 4. Tampilkan total info semua node pada BST */
    printf("\n");
    printf("\nSum of BST Info\t:  ");
    ...
    /* 5. Tampilkan banyaknya daun dari BST */
    printf("\nNumber of Leaves\t: ");
    ...
    /* 4. Tampilkan Tinggi dari Tree*/
    printf("\nHeight of Tree\t\t: ");
    ...
    return 0;
}
```

```
================================================================================
5 3 9 10 4 7 1 8 6

Pre Order            : 5 3 1 4 9 7 6 8 10

Descendent of Node 9 : 7 6 8 10

Sum of BST Info      : 53
Number of Leaves     : 5
Height of Tree       : 3
================================================================================
```

Gambar 3. Ilustrasi tampilan dari *main program*