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 NILAI JURNAL.
- Hanya MENGUMPULKAN tetapi TIDAK MENGERJAKAN = PENGURANGAN POIN.
- Deadline pengumpulan TP Modul 13 adalah Senin, 19 Desember 2022 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 : newNode_130121xxxx.
- File diupload di LMS menggunakan format PDF dengan ketentuan : TP_MODX_NIM_KELAS.pdf.

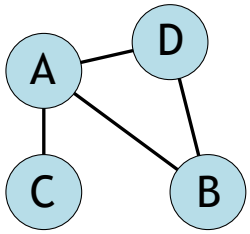
```
Contoh :
   adrNode newNode_130121xxxx (char x);
```

Contact Person :

- Rayhan Risq Arya B (WA : 082242147722)
- Shidqi Fadhlurrahman Yusri (WA : 081219461153)
- Hidayat Taufiqur Rahmah Achmad (WA : 081356636030)
- Syahdi Gharizah Ahsan (WA : 089691336972)
- Olaza Aurora Syafira (WA : 082352811045)

SELAMAT MENGERJAKAN ^^

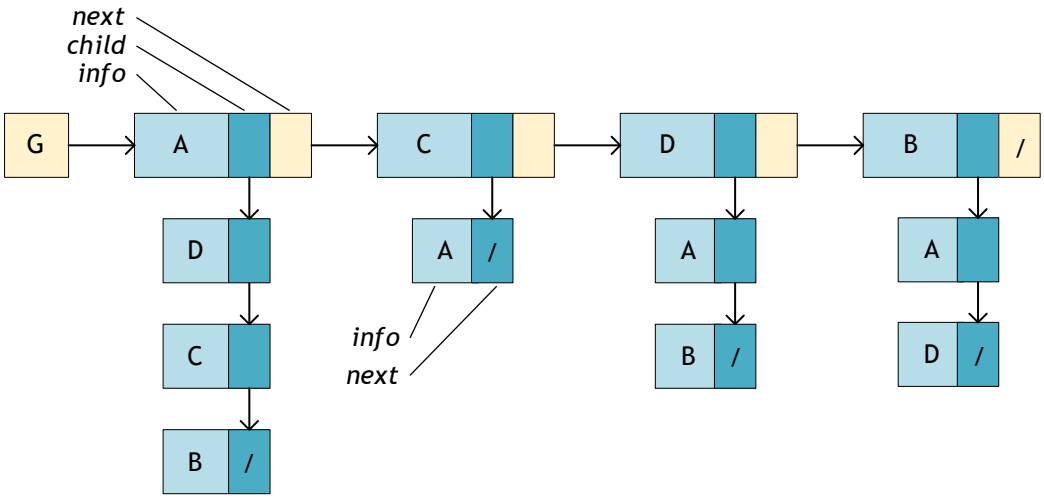
Tugas Pendahuluan Praktikum Modul 13
Struktur Data 2022/2023-1
"*Graph*"



Gambar 1. Sebuah contoh *graph* sederhana

Sebuah *graph* sederhana direpresentasikan oleh *multi linked-list*, di mana *list* utama atau *parent* merupakan *single linked-list* dari *node* yang terdapat di dalam *graph*, sedangkan *single linked-list* vertikal yang ada di setiap *node* menyatakan kumpulan *edge* dari *node* tersebut.

Elemen dari *node* berisi info dari *node*, *child* yang berisi alamat pertama dari *edge* pada *node* tersebut, dan *next* yang berisi alamat elemen *node* berikutnya pada *list*. Setiap elemen dari *edge* berisi dengan info tetangga dari *node* yang terdapat di dalam *list parent* serta *next* dari *list edge*.



Gambar 2. Ilustrasi *graph* pada Gambar 1 dalam *multi linked-list*

Soal Tugas Pendahuluan (Nilai Max : 10) :

1. (1 Poin) Buatlah ADT dari *graph* sederhana, asumsi info adalah sebuah *character* alphabet. Perhatikan tipe-tipe data yang digunakan pada nomor 2 untuk deklarasi *struct* pada file *header* graph.h . Kemudian,

2. Lengkapi primitif berikut ini pada file implementasi spesifikasi graph.cpp !

```
adrNode newNode(char x); (1 Poin)
/* mengembalikan alamat sebuah node baru dengan info berupa x */

void addNode(adrNode &G, adrNode p); (1 Poin)
/* I.S. terdefinisi alamat elemen pertama dari graph G (mungkin kosong), dan sebuah alamat dari node baru yang disimpan pada p.
   F.S. node baru ditambahkan ke dalam list parent sebagai elemen terakhir*/

adrNode findNode(adrNode G, char x); (1 Poin)
/* mengembalikan alamat node dengan info x pada graph G, atau NULL apabila tidak ditemukan */

void addEdge(adrNode &G, char x, char y); (2 Poin)
/* I.S. terdefinisi alamat elemen pertama dari graph G (mungkin kosong), dan character x dan y.
   F.S. node dengan info x dan y terhubung oleh sebuah edge, edge ditambahkan di awal pada list */

bool isConnected(adrNode G, char x, char y); (2 Poin)
/* mengembalikan true apabila x dan y terhubung oleh sebuah edge, atau false apabila tidak terhubung */

void printGraph(adrNode G); (1 Poin)
/* I.S. terdefinisi alamat elemen pertama dari graph G (mungkin kosong).
   F.S. menampilkan adjacency graph, perhatikan contoh yang diberikan */
```

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

```
int main(){
    adrNode G;
    // tambahkan node A, B, C dan D
    ...
    ...
    ...
    ...

    // tambahkan edge pada graph seperti Gambar 1
    ...
    ...
    ...
    ...

    // tampilkan graph seperti Gambar 3
    ...
    return 0;
}
```

```
================================================================================
node A: - C - D - B
node B: - A - D
node C: - A
node D: - B - A
================================================================================
```

Gambar 3. Ilustrasi tampilan dari *main program*