Topik: List Linier

Tujuan Praktikum:
Mahasiswa memahami implementasi dan penggunaan ADT List Linier dengan representasi berkait menggunakan pointer dalam bahasa C.

PETUNJUK PRAKTIKUM:
1. Setiap ADT dibuat dengan format penamaan file sebagai berikut:
   a. Untuk file header : <namaADT>.h
   b. Untuk file realisasi : <namaADT>.c

   Dengan:
   - <namaADT> : nama ADT dalam satu kata, contoh: point

   Contoh:
   point.h; point.c; mpoint.c
2. Untuk setiap file yang Anda buat, buat header sebagai berikut:

```
/* NIM/Nama  :                                                    */
/* Nama file :                                                    */
/* Topik     :                                                    */
/* Tanggal   :                                                    */
/* Deskripsi :                                                    */
```

3. Untuk setiap ADT, di-upload setelah dikompres menjadi 1 file dengan nama: p07<nim><namaADT>.zip.
   dengan:
   - <nim> : NIM Anda
   - <namaADT> : nama ADT dalam satu kata, contoh: point

   Contoh: p0713512500point.zip
4. Softcopy materi kuliah dan diktat, termasuk yang terkait dengan pemrograman dengan Bahasa C dapat dilihat pada situs **http://kuliah.itb.ac.id** pada link **IF2110/Algoritma dan Struktur Data**.
5. HANYA ADT YANG DAPAT DI-COMPILE YANG AKAN DIPERIKSA. File yang tidak dapat di-*compile* akan otomatis mendapatkan nilai 0.
6. Tugas ini bersifat INDIVIDUAL. Tidak ada toleransi bagi pencontek. Jika terbukti, baik yang dicontek maupun yang mencontek akan mendapatkan nilai 0.
7. Ikuti petunjuk asisten untuk pengumpulan tugas.

SELAMAT BEKERJA.

Soal 1. ADT List Linier (Bobot : 100%)

a. Ambillah ADT List Linier dengan representasi fisik berkait dengan pointer yang telah Anda kerjakan sebagai tugas pra-praktikum.
b. Pada ADT List Linier tersebut tambahkanlah fungsi dan prosedur berikut ini (kerjakan yang Anda anggap mudah terlebih dahulu):

1. Fungsi **CountX**

```
int CountX (List L, Infotype X, char sign);
/* Menghasilkan banyaknya elemen list L yang memenuhi
persyaratan
   sebagai berikut:
   - Jika sign adalah '>' maka dihasilkan banyaknya elemen L
     yang lebih besar dari X
   - Jika sign adalah '=' maka dihasilkan banyaknya bilangan
     yang sama dengan X
   - Jika sign adalah '<' maka dihasilkan banyaknya elemen L
     yang bernilai < X  */
```

2. Prosedur **FilterList**:

```
void FilterList (List * L, Infotype X);
/* IS : L terdefinisi, mungkin kosong. X terdefinisi. */
/* FS : Semua elemen L dengan info bernilai X dihapus dan
didealokasi
        dari list L. L mungkin menjadi kosong. */
```

3. Prosedur **AscOrderList**:

```
void AscOrderList (List * L);
/* IS : L terdefinisi, tidak kosong */
/* FS : Elemen-elemen L terurut membesar */
```

Petunjuk : Cara mengurutkan elemen bebas. Salah satu cara yang bisa dipakai adalah dengan *selection (minimum) sort* yang idenya adalah mencari nilai minimum antara sekelompok elemen list L lalu menukarkan info elemen pertama dari kelompok elemen tersebut dengan info elemen yang memiliki nilai minimum, dst.

4. Prosedur **SplitList**

```
void SplitList (List L, List * LGenap, List * LGanjil);
/* IS : L mungkin kosong, L1 dan L2 sembarang */
/* FS : Berdasarkan L, dibentuk dua buah list LGenap dan
LGanjil. LGenap berisi semua elemen L yang genap, sedangkan
LGanjil berisi semua elemen L yang ganjil. Setiap elemen LGenap
dan LGanjil harus dialokasi sebagai elemen baru. Jika ada
elemen yang gagal dialokasi, maka list yang bersangkutan
menjadi kosong dan semua elemen yang sudah telanjur dialokasi,
didealokasi (bisa LGenap atau LGanjil, atau dua-duanya). L
tetap.

Perhitungan elemen L, dimulai dari 1 (elemen pertama L
merupakan elemen ke-1)
*/
```

5. Fungsi **isSubList**

```
boolean isSubList (List L1, List L);
/* Menghasilkan true jika list L1 muncul sebagai sublist dari
L. L1 merupakan sublist dari L bila L1 muncul pada L secara
sekuensial.
   Contoh: isSubList([4,5,6],[3,4,5,6,7]) = true
           isSubList([4,5,6],[3,4,1,5,6]) = false
           isSubList([4,5,6],[4,5]) = false
```

```
            isSubList([1,2,3],[1,2,3,7]) = true
            isSubList([],[4,6,8]) = true
            isSubList([], []) = true
            isSublist([4,5,6], []) = false*/
```

6. Fungsi **Average**

```
double Average (List L);
/* Mengirimkan nilai rata-rata list L, rata-rata list kosong
adalah 0 */
```

7. Prosedur **Concat1**

```
void Concat1 (List *L1, List *L2, List *L3);
/* I.S. L1 dan L2 sembarang
   F.S. L1 dan L2 kosong, L3 adalah hasil konkatenasi L1 dan L2
   Konkatenasi dua buah list: L1 dan L2 menghasilkan L3 yang
baru (dengan elemen
   list L1 dan L2) dan L1 serta L2 menjadi list kosong.
   Tidak ada alokasi/dealokasi pada prosedur ini
*/
```

Program Akan **diuji menggunakan driver asisten**. Agar tidak terjadi *compile error*, semua *prototype* fungsi harus tersedia pada header(*.h) dan realisasi (*.c) walaupun *body*-nya kosong (seperti PraPraktikum).

Kumpulkan file (**listlinear.h listlinear.c**) dalam 1 file berekstensi *.zip