Topik: List Rekursif

Tujuan Praktikum:
Mahasiswa memahami implementasi dan pengelolaan list dengan pointer secara rekursif 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
   c. Untuk file driver/program utama : m<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: p10<nim><namaADT>.zip.
   dengan:
   - <nim> : NIM Anda
   - <namaADT> : nama ADT dalam satu kata, contoh: point

   Contoh: p1013514500point.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. Standar *compiler* yang digunakan untuk penilaian adalah *compiler* di lingkungan Linux.
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 List Rekursif

Ambillah ADT List sirkuler dengan representasi fisik berkait dengan pointer yang telah Anda kerjakan sebagai tugas pra-praktikum.
Tambahkan fungsi dan prosedur berikut ini ke ADT tersebut. Silahkan menambahkan fungsi/prosedur lain, jika diperlukan.

1. Fungsi IsOneElmt

```
function IsOneElmt (L : List) → boolean
{ Menghasilkan true jika L hanya terdiri dari satu elemen. }
```

2. Prosedur MinMax

```
procedure MinMax (input L : List, output Min, Max : Infotype)
{ I.S. L terdefinisi, Min dan Max sembarang.
  F.S. Min berisi nilai elemen L terkecil, Max berisi nilai elemen L terbesar. }
```

3. Fungsi ListCompare

```
function ListCompare (L1 : List, L2 : List) → integer
{ Menghasilkan 1 jika L1 > L2, 0 jika L1 = L2, dan -1 jika L1 < L2.
  L1 dan L2 tidak kosong.
  L1 = L2 jika memiliki jumlah elemen yang sama dan semua elemen pada urutan yang
          bersesuaian juga sama. Misalnya [3,5,7,2,8] = [3,5,7,2,8].
  L1 > L2 jika ditemukan e1x > e2x dengan x adalah posisi terkecil kemunculan
          elemen di L1 dan L2 yang tidak sama. ATAU jika semua character pada
          urutan yang bersesuaian sudah sama, tetapi panjang L1 > panjang L2.
          Misalnya [3,5,7,9] > [3,5,6,7,8]; [1,2,3,4,5] > [1,2,3].
  L1 < L2 adalah kebalikan dari L1 > L2. }
```

4. Prosedur SplitKelipatanX

```
procedure SplitOnX (input L : List, X : Infotype, output L1, L2 : List)
{ I.S. L dan X terdefinisi, L1 dan L2 sembarang.
  F.S. L1 berisi semua elemen L yang lebih kecil dari X, dengan urutan kemunculan
       yang sama, L2 berisi semua elemen L yang tidak masuk ke L1, dengan urutan
       kemunculan yang sama. }
```

5. Fungsi IsAllExist

```
function IsAllExist (L1, L2 : List) → boolean
{ Menghasilkan true jika semua elemen dalam L1 terdapat dalam L2 (tanpa
  memperhatikan urutan ataupun banyaknya elemen).
  Kedua list mungkin kosong. Jika L1 kosong, maka hasilnya false. }
```

6. Fungsi CopyReverse

```
function CopyReverse (L : List) → List
{ Menghasilkan list dengan elemen-elemen yang sama dengan L namun dengan urutan
  kemunculan yang terbalik.
  Misalnya: CopyReverse([2,3,4,5,6]) = [6,5,4,3,2] }
```