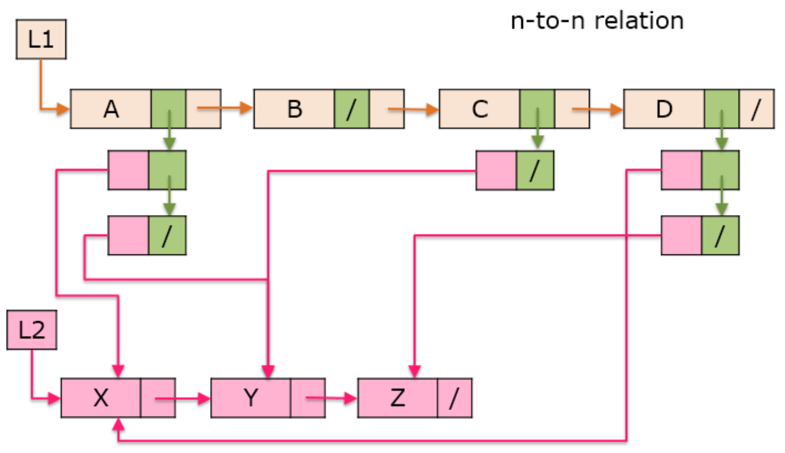
MULTI LINKED LIST

Multi Linked List -> bersifat induk dan anak (ada list anak di dalam sebuah elemen induk).

Contoh:

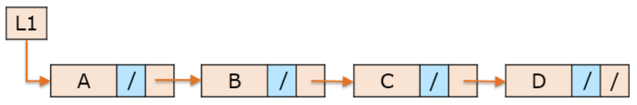
1. Setiap Mahasiswa mengambil 1 atau lebih matakuliah pada semester 1. (induk: mahasiswa, anak: matakuliah)
2. Student-Course (Relasi n to n)



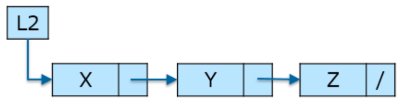
Gambar 1. Multi Linked List Student Course

Operasi/aksi yang dapat dilakukan:

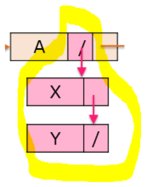
1. Insert dan Delete List Student (L1) -> sebagai Induk
2. Insert dan Delete List Anak (L3) -> sebagai Anak (Relasi antara List 1 dan List 2)
3. Insert dan Delete List Course (L2)



Gambar 2. L1 (List Student) yang berperan sebagai Induk



Gambar 3. L2 (List Course)



*Gambar 4. L3 (List relasi antara Student dan Course) *yang dilingkari garis kuning*

1. Deklarasi Struktur Data

```
Type infotype_Student<
        Id          : String
        Name        : String
>
Type infotype_course<
        Id          : String
        Course_name     : String
>
Type adr_student        : pointer to elm_student
Type adr_course         : pointer to elm_course
Type adr_anak           : pointer to elm_anak
Type elm_student<
        Info            : infotype_Student
        nextAnak        : listAnak              ...........*tadi dikelas pakenya adr_anak
        Next            : adr_student
>
Type elm_course<
        Info    : infotype_course
        Next    : adr_course
>
Type elm_anak<
        Info    : adr_course
        Next    : adr_anak
>
Type ListStudent< first: adr_student>
Type ListCourse< first: adr_course>
Type ListAnak< first: adr_anak>
L1: ListStudent
L2: ListCourse
L3: ListAnak
```

2. Search, Insert dan Delete List Student (L1)

```
Function SearchStudent(I: L: ListStudent, StudentID: String)→adrs_student
KD
        P: adr_student
        Status: true
Algoritma
        P← first(L)
        While (p<>nil) and (status=false) do
                If (info(P).Id = StudentID) then
                        Status← true
                Else
                        P← next(P)
        → P
```

```
Procedure InsertFirstStudent(I/O L:ListStudent, x: infotype_Student)
KD
    P: adr_student

Algoritma
        Alokasi(P)
        Info(P)← x
        nextAnak(P)← nil
        if (first(L)=nil) then
                first(L)← P
        else
                next(P)← first(L)
                first(L)← P
```

```
Procedure InsertLastStudent(I/O L: ListStudent, I x: infotype_Student)
KD
    P: adr_student
    Temp: ade_student

Algoritma
        Alokasi(P)
        Info(P)← x
        nextAnak(P)<- nil
        next(P)← nil
        If (first(L)=nil) then
                first(L)← P
        else
                Temp← first(L)
                While (next(Temp)<>nil) do
                        Temp← next(Temp)
                next(Temp)← P
```

```
Procdure DeleteFirstStudent(I/O L: ListStudent)
KD
    P: adr_student
Algoritma
    P← first(L)
    first(L)← next(P)
    next(P)← nil
    dealokasi(P)
```

```
Procedure DeleteLastStudent(I/O L:ListStudent)
KD
    P: adr_student
    Q: adr_student
Algoritma
    P← first(L)
    While (next(P)<>nil) do
        Q← P
        P← next(P)
    next(Q) ← nil
    dealokasi(P)
```

```
Procedure DeleteByIDStudent(I/O L: ListStudent, StudentID: String)
KD
    P: adr_student
    Q: adr_student
Algoritma
    P← SearchStudent(L,StudentID)
    If (P<>nil) then
        If (P=first(L)) then
            deleteFirstStudent(L)
        else
            if (next(P)=nil) then
                deleteLastStudent(L)
            else
                Q← first(L)
                While (next(Q)<>P) do
                    Q← next(Q)
                next(Q)← next(P)
                next(P)← nil
                dealokasi(P)
```

3. Insert dan Delete List Course (L2)

```
Function SearchCourse(I: L: ListCourse CourseID: String)→adrs_course
KD
        P: adr_course
        Status: true
Algoritma
        P← first(L)
        While (p<>nil) and (status=false) do
                If (info(P).Id = CourseID) then
                        Status← true
                Else
                        P← next(P)
        → P
```

Procedure **InsertFirstCourse**(I/O L:ListCourse, x: infotype_course)
KD   P: adr_course

Algoritma
    Alokasi(P)
    Info(P)← x
    if (first(L)=nil) then
        first(L)← P
    else
        next(P)← first(L)
        first(L)← P

Procedure **InsertLastCourse**(I/O L: ListCourse, I x: infotype_course)

KD
    P: adr_course
    Temp: adr_course

Algoritma
    Alokasi(P)
    Info(P)← x
    next(P)← nil
    If (first(L)=nil) then
        first(L)← P
    else
        Temp← first(L)
        While (next(Temp)<>nil) do
            Temp← next(Temp)
        next(Temp)← P

```
Procdure DeleteFirstCourse(I/O L: ListCourse)
KD
    P: adr_course
Algoritma
    P← first(L)
    first(L)← next(P)
    next(P)← nil
    dealokasi(P)
```

```
Procedure DeleteLastCourse(I/O L:ListCourse)
KD
        P: adr_course
        Q: adr_course
Algoritma

        P← first(L)
        While (next(P)<>nil) do
                Q← P
        P← next(P)
        next(Q) ← nil
        dealokasi(P)
```

```
Procedure DeleteByIDCourse(I/O L: ListCourse, CourseID: String)
KD
    P: adr_course
    Q: adr_course
Algoritma
    P← SearchCourse(L,CouseID)
    If (P<>nil) then
            If (P=first(L)) then
                    deleteFirstCourse(L)
            else
                    if (next(P)=nil) then
                            deleteLastCourse(L)
                    else
                            Q← first(L)
                            While (next(Q)<>P) do
                                    Q← next(Q)
                            next(Q)← next(P)
                            next(P)← nil
                            dealokasi(P)
```

4. Insert dan Delete List Anak (L3)

```
Function SearchAnak(I: L: ListAnak, CourseID: String)→adrs_anak
KD
        P: adr_anak
        Status: true
Algoritma
        P← first(L)
        While (p<>nil) and (status=false) do
                If (info(P).info.Id = CourseID) then
                        Status← true
                Else
                        P← next(P)
        → P
```

Procedure **InsertFirstAnak**(I/O L:ListAnak, x: adr_course)
KD   P: adr_anak

Algoritma
    Alokasi(P)
    Info(P)← x
    if (first(L)=nil) then
        first(L)← P
    else
        next(P)← first(L)
        first(L)← P

Procedure **InsertLastAnak**(I/O L: ListAnak, x: adr_course)

KD
    P: adr_anak
    Temp: adr_anak

Algoritma
    Alokasi(P)
    Info(P)← x
    next(P)← nil
    If (first(L)=nil) then
        first(L)← P
    else
        Temp← first(L)
        While (next(Temp)<>nil) do
            Temp← next(Temp)
        next(Temp)← P

```
Procdure DeleteFirstAnak(I/O L: ListAnak)
KD
    P: adr_anak
Algoritma
    P← first(L)
    first(L)← next(P)
    next(P)← nil
    dealokasi(P)
```

```
Procedure DeleteLastAnak(I/O L:ListAnak)
KD
        P: adr_anak
        Q: adr_anak
Algoritma

        P← first(L)
        While (next(P)<>nil) do
                Q← P
        P← next(P)
        next(Q) ← nil
        dealokasi(P)
```

```
Procedure DeleteAnakByCourse(I/O L: ListAnak, CourseID: String)
KD
        P: adr_course
        Q: adr_course
Algoritma
        P← SearchAnak(L,CouseID)
        If (P<>nil) then
                If (P=first(L)) then
                        deleteFirstAnak(L)
                else
                        if (next(P)=nil) then
                                deleteLastAnak(L)
                        else
                                Q← first(L)
                                While (next(Q)<>P) do
                                        Q← next(Q)
                                next(Q)← next(P)
                                next(P)← nil
                                dealokasi(P)
```

5. Insert anak (Add course) ke induk (L1)
   Procedure addCourse(i/o L1:listStudent, i: L2: listCourse, id: String, courseID: String)

```
KD
        P: adr_student
        Q: adr_course
        R: adr_relation
        L3: listAnak
Algoritma
        P← search_student(L1, id)
        Q← search_course(L2, courseID)
        If (p<>nil) and (q<>nil) then
                Alokasi(Z)
                Info(Z)← Q
                L3← nextAnak(P)
                insertLastAnak(L3,R) ............boleh diganti insert last/after.
```

   ***catatan: dicoba untuk kasus delete Course, maka data list anak yang mengandung Course tersebut juga dihapus. Semangat ☺**