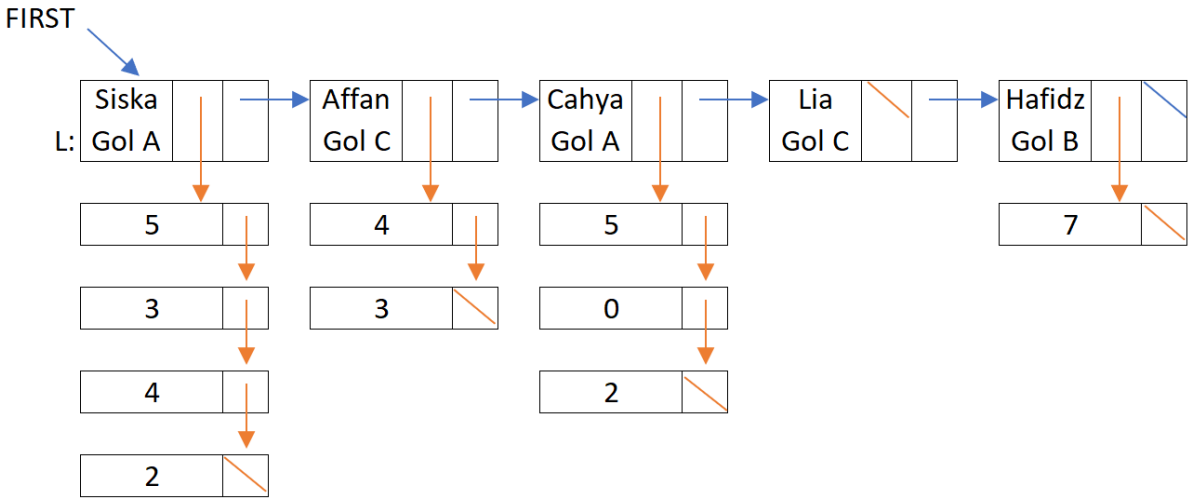
JURNAL MODUL 10 - STRUKTUR DATA

MULTI LINKED LIST

Jurnal Modul 10 merupakan Jurnal Terbimbing



multi.h (8 Poin)

```
Type adr_sales : <pointer to elm_sales>
Type adr_jual : <pointer to elm_jual>
Type sales {string nama, gol}
Type elm_sales : <info : sales,
                  Next : adr_sales
                  nextJual : adr_jual>
Type elm_jual : <info : int,
                 Next : adr_jual>

Type mll : <first : adr_sales>

Procedure Create_list(input/output List_Sales : mll)
Procedure newElm_Sales(input info : sales, input/output S : adr_sales)
Procedure newElm_Jual(input info : integer, input/output J : adr_jual)
Procedure insertNew_Penjualan(input/output List_Sales : mll,  input  S : adr_sales, J : adr_jual)
Procedure insertLast_Sales(input/output List_Sales : mll, input S : adr_sales)
Procedure deleteFirst_Penjualan(input/output List_Sales : mll, input S : adr_sales, J : adr_jual)
Procedure deleteLast_Penjualan(input/output List_Sales : mll, input S : adr_sales, output J : adr_jual)
Procedure deleteAfter_Penjualan(input/output List_Sales : mll, input S : adr_sales, prec : adr_jual,
output J : adr_jual)
Procedure Delete_Penjualan(input/output  List_Sales : mll)
Procedure showData_Sales(input List_Sales : mll)
Function Search_Sales(List_Sales : mll, input nama_Sales : string) →adr_sales
```

multi.cpp

Asumsi Fungsi dan Prosedur di bawah ini sudah terdefinisi, Anda bisa langsung menggunakannya (Total 12 Poin)

Function Search_Sales(List_Sales : mll, **input** nama_Sales : **string**) →adr_sales

{Fungsi ini akan mencari nama sales pada list sales, jika ditemukan maka akan dikembalikan alamatnya, jika tidak ditemukan maka akan dikembalikan NIL}

```
adr_sales Search_Sales(mll List_Sales, string nama_Sales){
    adr_sales P = first(List_Sales);
    while (P != NULL){
       if(info(P).nama == nama_Sales){
            return P;
       }
       P = next(P);
    }
    return NULL;
}
```

Procedure insertLast_Sales(**input/output** List_Sales : mll, **input** S : adr_sales)

{I.S. Terdefinisi list sales yang mungkin kosong, dan elemen sales baru yang disimpan oleh pointer S, yang akan diinsertkan menjadi elemen sales paling akhir pada list sales
F.S. elemen baru menjadi elemen sales paling akhir pada list sales}

```
void insert_last_Sales(mll &List_Sales, adr_sales S){
    if(first(List_Sales) == NULL){
        first(List_Sales) = S;
    }else{
        adr_sales P = first(List_Sales);
        while (next(P) != NULL){
            P = next(P);
        }
        next(P) = S;
    }
}
```

Procedure deleteLast_Penjualan(**input/output** List_Sales : mll, **input** S : adr_sales, **output** J : adr_jual)

{I.S. Terdefinisi list sales yang tidak kosong, dan elemen sales yang akan dihapus elemen penjualannya yang berada di posisi paling akhir
F.S. Elemen penjualan paling akhir dari sales S dihapus dan alamatnya disimpan di pointer J.
*CLUE : **Konsep delete last** }*

```
void Delete_Last_Penjual(mll &List_Sales, adr_sales &S, adr_jual &J){
    adr_jual P = nextJual(S);
    while (next(P) != J){
        P = next(P);
    }
    J = next(P);
    next(P) = next(J);
    next(J) = NULL;
}
```

Procedure deleteAfter_Penjualan(**input/output** List_Sales : mll, **input** S : adr_sales, prec : adr_jual, **output** J : adr_jual)

{I.S. Terdefinisi list sales yang tidak kosong, pointer prec
F.S. Elemen penjualan yang berada setelah pointer prec dihapus.
*CLUE : **Konsep delete after**}*

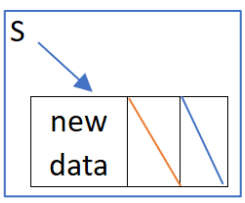
```
void Delete_after_Penjualan(mll &List_Sales, adr_sales &S, adr_jual prec, adr_jual &J){
    next(prec) = next(J);
    next(J) = NULL;
}
```

Procedure Create_list(**input/output** List_Sales : mll) **(5 Poin)**

{I.S. –
F.S. Dihasilkan sebuah multi linked list 1 ke N dengan pointer first yang NIL}
Kamus Data
Algoritma
// NIL kan pointer First dari list

Procedure newElm_Sales(**input** info : sales, **input/output** S : adr_sales) **(6 Poin)**

{I.S. terdefinisi data sales dan pointer yang akan menyimpan alamat elemen dari data sales yang baru
F.S. data sales baru sudah menjadi elemen dan alamatnya disimpan oleh pointer S}
Kamus Data
Algoritma
//alokasi elemen sales yang alamatnya disimpan oleh pointer S
……
//NIL kan pointer next dari elemen
……
//NIL Kan pointer next_jual dari elemen
……
//masukkan data sales ke dalam info elemen
……
……



Procedure newElm_Jual(**input** info : **integer**, **input/output** J : adr_jual) **(6 Poin)**

{I.S. terdefinisi data jumlah penjualan dan pointer yang akan menyimpan alamat elemen dari data penjualan yang baru

F.S. data sales baru sudah menjadi elemen dan alamatnya disimpan oleh pointer S}

Kamus Data

ALgoritma

//alokasi elemen jual yang alamatnya disimpan oleh pointer J

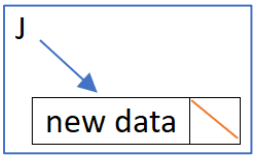
……

//NIL kan pointer next dari elemen

……

//masukkan data penjualan ke dalam info elemen

……

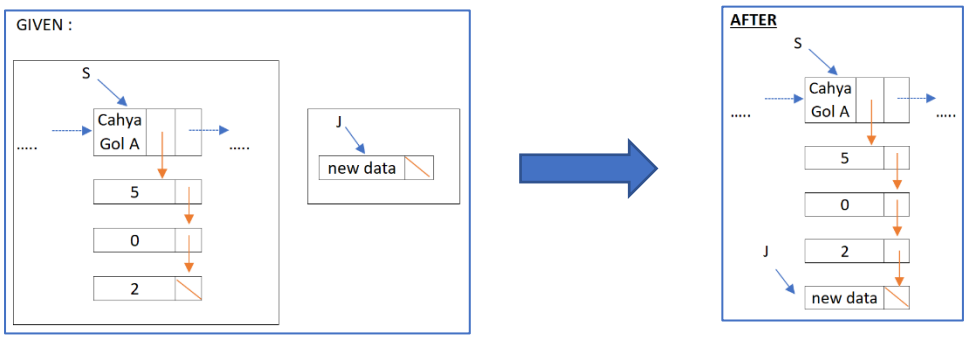


Procedure insertNew_Penjualan(**input/output** List_Sales : mll, **input** S : adr_sales, J : adr_jual) **(12 Poin)**

{I.S. Terdefinisi list sales yang tidak kosong, dan elemen penjualan baru yang akan disisipkan menjadi data penjualan pada sales S. Data penjualan pada sales S tidak kosong.

*F.S. Elemen penjualan baru akan disisipkan menjadi data penjualan **paling akhir** di sales pada elemen S. CLUE : **Konsep Insert Last**}*

ILUSTRATION



Kamus Data

……

Algoritma

// cari lokasi elemen penjualan paling akhir

……

……

……

……

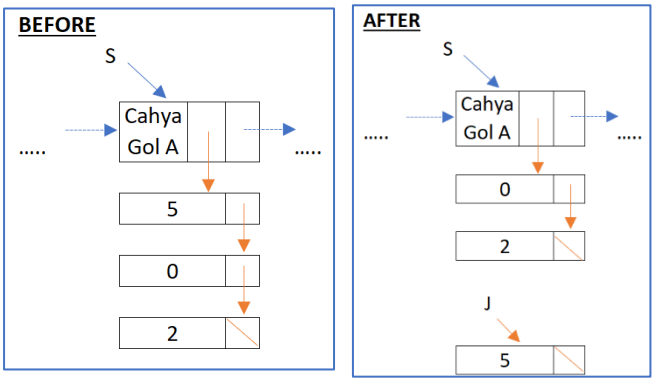
//sambungkan elemen paling akhir ke elemen yang baru

……

Procedure deleteFirst_Penjualan(**input/output** List_Sales : mll, **input** S : adr_sales, J : adr_jual) **(8 Poin)**

{I.S. Terdefinisi list sales yang tidak kosong, dan elemen sales yang akan dihapus elemen penjualannya yang berada di posisi paling pertama

F.S. Elemen penjualan pertama dari sales S dihapus dan alamatnay disimpan di pointer J. CLUE : ***Konsep delete first*** *}*



Kamus

Algoritma

……

Procedure showData_Sales(**input** List_Sales : mll) **(12 Poin)**

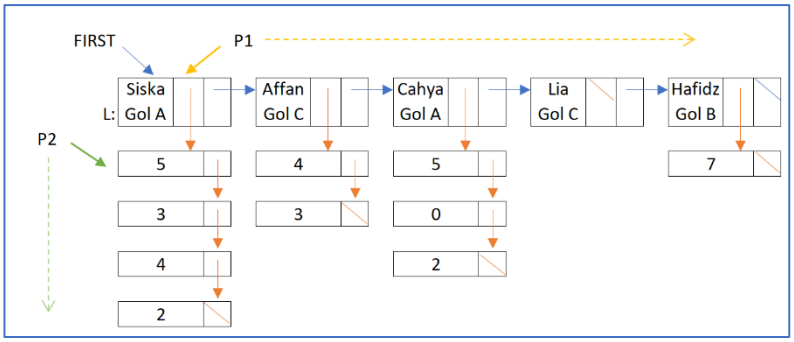
{I.S. Terdefinisi list sales yang tidak kosong

F.S. Data penjualan setiap sales tampil ke layar}

Tampilan :

Siska
5 3 4 2
Affan
4 3
Cahya
5 0 2
Lia
Hafidz
7

HINT : Anda membutuhkan 2 pointer, pointer yang akan menelusuri elemen sales satu persatu, dan setiap kali pointer itu mengunjungi sebuah elemen sales, ada pointer kedua yang akan menelusuri elemen data penjualan dari sales tsb. Setelah selesai penelusuran elemen penjualan di sebuah elemen sales, maka pointer pertama tadi bergeser ke sales selanjutnya, dna mengulangi Langkah yang sama, hingga semua sales terkunjungi.



Kamus Data

……

Algoritma

// setting awal pointer sales

……..

//loop pointer sales menelusuri semua elemen sales

……

//mengoutputkan info sales

……

//setting start pointer data penjualan di elemen penjualan pertama dari elemen sales saat ini

……..

//Loop pointer data penjualan menelusuri semua data penjualan dari seles saat ini

……..

//mengoutputkan info penjualan

……

//pindah ke data penjualan berikutnya

……

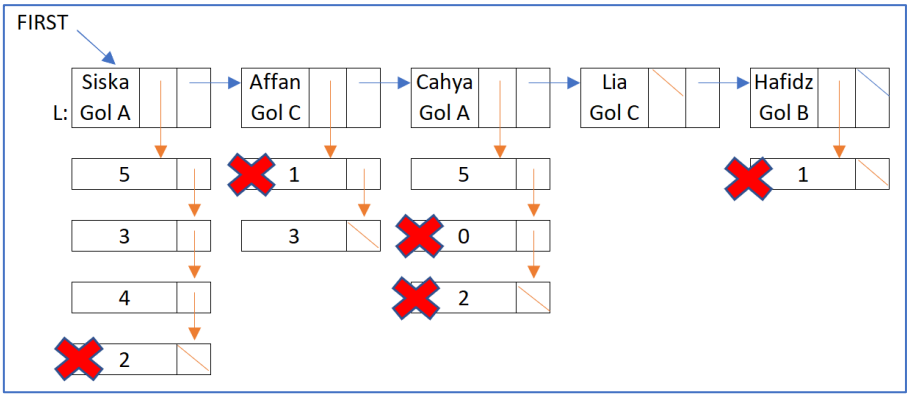
//pindah ke data sales berikutnya

……

Procedure Delete_Penjualan(**input/output** List_Sales : mll) **(20 Poin)**

{I.S. Terdefinisi list sales yang tidak kosong

F.S. Data penjualan kurang dari 3 dihapus dari elemen penjualan}



Clue : telusuri semua data penjualan dari setiap sales, jika menenukan data penjualan kurang dari 3 maka lakukan penghapusan dengan cara memanggai procedure delete sesuai kondisi elemennya (Apakah elemen nya di paling awal, atau di paling akhir, atau diantaranya)

Kamus Data

//fungsi procedure ini sudah terdefinisi, Anda tinggal panggil saja sesuai kebutuhan
Procedure Delete_First_Penjualan (In/Out List_Sales : mll, In S: adr_sales, out J : adr_jual)
Procedure Delete_Last_Penjualan (In/Out List_Sales : mll, In S: adr_sales, out J : adr_jual)
Procedure Delete_after_Penjualan (In/Out List_Sales : mll, In S: adr_sales, In prec: adr_jual, out J : adr_jual)

Algoritma

```
// setting awal pointer sales
……..
//loop pointer sales menelusuri semua elemen sales
……

    //setting start pointer data penjualan di elemen penjualan pertama dari elemen sales saat ini
    ……..

    //Loop  pointer data penjualan menelusuri semua data penjualan dari seles saat ini
    ……..

        //Pengecekan apakah data penjualan saat ini kurang dari 3 atau tidak.
        …….
            //Action ketika kurang dari 3 adalah mengecek apakah elemen nya berada di data
            paling pertama di sales tersebut, jika ya, lakukan delete first, dst
            IF (……) THEN
                ……
            ELSE IF (……) THEN
                ……
            ELSE
                ……..

        //berpindah ke data penjualan berikutnya
        ……
    //berpindah ke data sales berikutnya
    ……
```

Buatlah program utama (main.cpp) untuk menguji implementasi Multi Linked List (10 Poin)