**Nama: Rafid Damar Pratama  
NIM : 1201230026   
KELAS : SE – 06 - 01  
  
  
  
Capaian Pembelajaran**: Mahasiswa mampu mengimplementasikan Double Linked List.

**Pokok Bahasan**: Double Linked List

**Tujuan Tugas**: Implementasi Double Linked List dengan menggunakan notasi algoritma

**Aturan**:

1. Kerjakan secara individu
2. Jawaban ditulis di kertas kemudian difoto dan diupload di LMS

**Jumlah Soal**: 1 Soal

**SOAL 1**

Untuk menyimpan data akun yang registrasi ke aplikasi X digunakan representasi Linear Double Linked List. Data akun yang disimpan terdiri dari **username dan password**. Username harus UNIK. Berikut ilustrasinya:

Last

First

xyz

001pqr

userAbc

123abc

user1

password

1. Buat definisi struktur datanya (ADT)!  
   Type infotype : objMember {username;password}  
   type addres : pointer to objMember  
     
   type objMember <  
    info : infotype  
    next : addres   
    prev : addres   
   >  
     
   Type List : <  
    First : addres   
    Last : addres   
   >  
     
   Kamus :  
   L : List   
   P : Pointer to objMember
2. Buat implementasi fungsi/ prosedur berikut untuk melakukan registrasi/ sign up akun:
3. function createNewElm (akun: infotype) 🡪 address

*{Membuat (mengalokasikan) sebuah elemen baru. Jika alokasi berhasil, info dari elemen berisi akun, prev dan next dari elemen berisi nil. Kembalikan (return) address dari elemen baru tsb.}  
  
x : Elemen baru   
  
  
Alokasi (P)  
info(p) 🡸 x  
next(p) 🡸 null  
prev(p) 🡸 null   
 return p*

1. procedure InsertLast (In p: address; In/Out L: List)

*{Menambahkan sebuah elemen baru beralamat p sebagai elemen terakhir list L.*

*IS: List tidak kosong, p sudah dialokasi, p≠nil, next(p)=nil.*

*FS: p adalah elemen terakhir list L.}  
  
x : new elemen masukan   
prec : x  
prec : pointer to addres*

*P : Last(L)  
 prev(prec) 🡸 p  
 next(p) 🡸 prec  
 Last(L) 🡸 prec*

1. function findAkun (username: String, L: List) 🡪 address

*{Mencari akun dengan username tertentu.*

*IS: Terdefinisi List L dan username yang akan dicari. List mungkin kosong.*

*FS: Return alamat dimana akun ditemukan, return nil jika tidak ditemukan. }  
  
Allocate(p)   
p 🡸 Fisrt(L)  
x : username yang dicari  
int idx = 0   
  
if (info(Fisrt(L) = nil) and (next(p) = nil) :  
 return nil // List kosong   
else if (info(First(L) = x):  
 idx ++*

*return output( x ditemukan pada urutan ke {idx})  
else:  
 while next(p) != nil :  
 if (x = info(p)):  
 idx ++  
 return output( x ditemukan pada urutan ke {idx})*

*Else :   
 idx ++  
 p* 🡸 *next(p)  
return nil // jika returnoutput tidak dijalankan return ini dijalankan(tidak ditemukan)*

1. procedure signUp (In akun: infotype, In/Out L: List)

*{IS: Terdefinisi List L serta data akun baru. List mungkin kosong.*

*FS: Jika akun belum terdaftar, sisipkan sebagai elemen terakhir List L; jika sudah, tampilkan pesan “Account has been registered.” }  
  
Allocate(p)   
x : akun signUp  
p 🡸 x  
prec 🡸 first(L)  
bool cek : true  
if (info(Fisrt(L) = nil) and (next(prec) = nil) :  
 first(L) 🡸 p*

*Last(L) 🡸 p   
else :  
 while next(prec) != nil :  
 if (info(prec) = p):  
 output(“Account has been registered.”)  
 cek : false*

*Else :   
 prec 🡸 next(prec)  
if cek = true :  
 prev(p) 🡸 Last(L)  
 next(Last(L)) 🡸 p  
 Last(L) 🡸 p*

1. Buat implementasi prosedur berikut untuk men-remove sebuah akun:
2. procedure deleteFirst (In/Out L: List, Out p: address)

*{ IS: Terdefinisi List L yang memiliki lebih dari 1 elemen.*

*FS: p adalah elemen pertama yang dihapus dari List L. }  
  
 Allocate(p)  
p 🡸First(L)  
  
Fisrt(L)🡸 next(p)  
next(p)🡸 nil  
prev(First(L)) 🡸 nil*

1. procedure deleteAfter (In q: address, Out p: address)

*{ IS: Terdefinisi List L yang memiliki lebih dari 2 elemen., q ≠ nil, prev(q) ≠ nil, next(q) ≠ last.*

*FS: p adalah elemen setelah q yang dihapus dari List L. }  
  
p🡸 next(q)  
q 🡸 prev(p)  
//yang di delete p(after q)  
next(q) 🡸 next(p)  
prev(next(p))🡸 q  
//putuskan  
prev(p) 🡸 nil  
next(p) 🡸 nil*

1. procedure deleteLast (In/Out L: List, Out p: address)

*{ IS: Terdefinisi List L yang memiliki lebih dari 1 elemen.*

*FS: p adalah elemen terakhir yang dihapus dari List L. }  
  
Allocate(p)  
p 🡸 Last(L)  
Last(L) 🡸 prev(Last(L))  
// putuskan   
prev(p)🡸 nil   
next(p)🡸 nil*

1. procedure removeAkun (In username: String, In/Out L: List

*{ IS: List mungkin kosong. Jika tidak kosong, minimal memiliki 1 elemen.*

*FS: Akun dengan username tsb di-dealokasi dari List L.*

*Notes:*

* *Gunakan function b(iii) untuk mencari akun.*
* *Akun yang dihapus mungkin merupakan elemen pertama, tengah, atau terakhir.}*

*menggunakan b(iii) untuk mencari  
 p🡸 akun yang ingin di remove   
  
 //untuk elemen di awal  
 p 🡸 First(L)  
 Fisrt(L) 🡸 next(p)  
 next(p) 🡸 nil   
 prev(First(L) 🡸 nil   
 deallocate(p)  
  
 //untuk elemen di tengah  
 current : pointer to elmList   
 prev(p) 🡸 current //current berada di sebelum p  
 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
 next(current)🡸 next(p)  
 prev(next(p)) 🡸 current   
 prev(p) 🡸 nil  
 next(p) 🡸 nil  
  
 //untuk elemen akhir   
 p🡸 Last(L) //p di last(L)  
 Last(L) 🡸 prev(Last(L))  
 prev(p) 🡸 nil   
 next(Last(L)) 🡸 nil*