Function ListEmpty(Input L:List) -> boolean  
// Mengembalikan nilai true / false jika list masih kosong.  
Begin

return(First(L)==Nil)

End

Procedure CreateList(Input/Output L:List)  
// Membuat list dengan mengisi nilai null di next pertama  
Begin

First(L) <- Nil

End

Function Alokasi(Input X:infotype) -> address

// Mengalokasikan sebuah nilai di memory, jika berhasil akan mengembalikan address dari nilai tersebut

Begin

P : address

P = (address)malloc(sizeof(ElmtList))

if(P!=Nil) then

Info(P) <- X

Next(P) <- Nil

endif

return P

End

Procedure Dealokasi(Input/Output P : address )

// Mengosongkan nilai di memory yang berada di suatu address

Begin

Next(P) <- Nil

free(P)

End

Function Search1(Input L:List, X:infotype) -> address

// Mencari suatu nilai pada List, jika ada maka mengembalikan addressnya.

Begin

P : address  
 P <- First(L)

found : boolean  
 found <- false

while (P!=Nil) AND ( NOT found)

if(Info(P)==X) then  
 found <- true  
 else   
 P <- Next(P)  
 endif

endwhile

return P

}

Function Search2(Input L:List L, X:infotype) -> address

// Mencari suatu nilai di List jika ada mengembalikan address dari nilai tersebut

Begin

P : address

if(First(L)!=Nil) then

return Nil

else

P <- First(L)

while((Next(P)!=Nil) AND (Info(P)!=X)) {

P <- Next(P)

endwhile

if(Info(P)==X)then

return P

else

return Nil

endif

endif

return P

End

Function FSearch(Input L:List, P:address) -> boolean

// Mencari apakah ada elemen list yang memiliki address tertentu

Begin

PTemp : address  
 found : boolean

Ptemp <- First(L)

found <- false

while((PTemp!=Nil) AND (NOT found)){

if(PTemp==P) then

found <- true

else

Ptemp <- Next(PTemp)

endif

endwhile

return found

End

Function SearchPrec(Input L:List, X:infotype) -> address

// Mencari nilai pada List jika ada maka mengembalikan address sebelum address yang dicari

Begin

Prec : address

P : address

if(Next(P)==Nil) then

Prec <-Nil

else

P <- First(L)

while((Next(P)!=Nil) AND(Info(P)!=X))

Prec <- P

P <- Next(P)

}

if(Info(P)==X)then

return Prec

else

return Nil

endif

endif

End

Procedure InsVFirst(Input/Output L:List, Input X:infotype)

// Menambahkan elemen pertama di List dengan nilai X

Begin

P : address  
 P <- Alokasi(X)

if(P!=Nil) then

InsertFirst(&(L),P)

endif

End

Procedure InsVLast(Input/Output L:List, Input X:infotype)

// Menambahkan sebuah elemen berisi X pada akhir List

Begin

P : address   
 P <- Alokasi(X)

if(P!=Nil)then

InsertLast(&(L),P)

endif

End

Procedure DelVFirst(Input/Output L:List, Input/Output X:infotype)

// Menghapus elemen pertama List dan nilai indo disimpan pada X

Begin

address P

DelFirst(&(L),&P)

X <- Info(P)

Dealokasi(&P)

End

Void DelVLast(List \*L, infotype \*X)

//Menghapus elemen list terakhir dan menyimpan nilai indo di X

Begin

address P

DelLast(&(\*L),&P)

\*X <- Info(P)

Dealokasi(&P)

End

Procedure InsertFirst(Input/Output L:List, Input P:address)

// Menambah elemen ber-address P di awal List

Begin

Next(P) <- First(L)

First(L) <- P

End

Procedure InsertAfter(Input/Output L:List, Input P:address, Input Prec:address )

// Menambah elemen ditengah list sesudah elemen beralamat Prec

Begin

Next(P) <- Next(Prec)

Next(Prec) <- P

End

Procedure InsertLast(Input/Output L:List,Input address P)

// Menambahkan elemen terakhir

Begin

address Last

if(ListEmpty(L)) then

InsertFirst(&(L),P)

else

Last <- First(L)

while(Next(Last)!=Nil)

Last <- Next(Last)

endwhile

Next(Last)=P

endif

End

Procedure DelFirst (Input/Output L:List, Input/Output P:address)

// Menghapus elemen pertama dari List

Begin

P <- First(L)

First(\*L) <- Next(First(L))

Next(P) <- Nil

End

Procedure DelP(Input/Output L:List, Input X:infotype)

// Menghapus list yang memiliki info X

Begin

P : address

P <- First(L)

while((Next(P)!=Nil) AND (Info(P)!=X)){

P <- Next(P)

Endwhile

if(Info(P)==X) then

Dealokasi(&P)

endif

End

Procedure DelLast(Input/Output L:List, Input/Output P:address)

// Menghapus elemen terakhir dari List

Begin

Last,PrecLast : address

Last <- First(L)

if(Next(Last)==Nil) then

DelFirst(&(L),&(Last))

Else

PrecLast <- Nil

while(Next(Last)!=Nil)

PrecLast <- Last

Last <- Next(Last)

endwhile

\*P <- Last

Next(PrecLast) <- Nil

endif

End

Procedure DelAfter(Input/Output L:List, Input/Output Pdel:address , Input Prec:address)

// Menghapus elemen di tengah sebelum alamat Prec

Begin

Pdel <- Next(Prec)

Next(Prec) <- Next(Pdel)

}

Procedure PrintInfo(Input L:List )

// Menampilkan info elemen list pada layar

Begin

i : int

P : address

P <- First(L)

if( P==Nil) then

write(layar) "List Kosong !\n"

else

write(layar) "[ "

do

write(layar) Info(P)

P <- Next(P)

i++

while(P!=Nil)

write(layar) "]"

endif

write(layar) "\n"

End

Function NbElmt(Input L:List) -> int

// Mengembalikan jumlah element list

Begin

P:address

NbEl : int

NbEl <- 0

if(ListEmpty(L)) then

return 0

else

P <- First(L)

do

NbEl++

P <-Next(P)

while(P!=Nil)

endif

return NbEl

End

Function Max(Input L:List) -> infotype

// Mengembalikan nilai terbesar dari List

Begin

P : address

Mmax : infotype MMax

if(!ListEmpty(L)) then

P <- First(L)

Mmax <- Info(P)

while(Next(P)!=Nil)

P <- Next(P)

if(Info(P)>MMax)then

Mmax <- Info(P)

endif

endwhile

endif

return MMax

End

Function AdrMax(Input L:List) -> address

// Mengembalikan address elemen list yang memiliki nilai terbesar

Begin

Pmax,P : address

Max : infotype

if(!ListEmpty(L)) then

P <- First(L)

Max <- Info(P)

while(Next(P) != Nil) {

P <- Next(P)

if(Info(P)>Max) then

Max <- Info(P)

Pmax <- P

endif

endwhile

endif

return PMax

End

Function Min(Input L:List) -> infotype

// Mengembalikan nilai yang terkecil dari List

Begin

P : address

Mmin : infotype

if(!ListEmpty(L)) then

P <- First(L)

Mmin <- Info(P)

while(Next(P)!=Nil)

P <- Next(P)

if(Info(P)< MMin) then

Mmin <- Info(P)

endif

endwhile

endif

return MMin

End

Function AdrMin(Input L:List) -> address

// Mengembalikan address elemen list yang memiliki nilai terkecil

Begin

PMin,P : address

Min : infotype

if(!ListEmpty(L)) then

P <- First(L)

Min <- Info(P)

while(Next(P) != Nil)

P <- Next(P)

if(Info(P)<Min) then

Min <- Info(P)

Pmin <- P

endif

endhile

endif

return PMin

End

Function Average(Input L:List) -> float

// Mengembalikan nilai rata-rata dari List

Begin

P : address

NbEl,Count : infotype

avg : float

if(!ListEmpty(L)) then

P <- First(L)

Count <- Info(P)

NbEl <- 1

while(Next(P)!=Nil)

P <- Next(P)

Count <- Count+Info(P)

NbEl++

endhile

avg <- (float)Count/(float)NbEl

endif

return avg

End

Procedure DelAll(Input/Outpur L:List)

// Menghapus semua elemen dari elemen pertama

Begin

X : infotype

while(!ListEmpty(L))

DelVFirst(&(L),&X)

endwhile

End

Procedure InversList(Input/Output L:List)

// Membalik elemen pertama jadi terakhir dan seterusnya

Begin

Ltemp : List

P : address

CreateList(&LTemp)

while(First(L)!=Nil)

DelFirst(&(L),&P)

Next(P) <- Nil

InsertFirst(&LTemp,P)

endwhile

First(L) <- First(LTemp)

End

Function FInversList(Input L:List) -> List

// Mengembalikan list baru yang berisi nilai invers dari L

Begin

LB : List

P,PB : address

X : infotype

P <- First(L)

CreateList(&LB)

do {

X <- Info(P)

PB <- Alokasi(X)

if(PB!=Nil) then

InsertLast(&LB,PB)

P <- Next(P)

else

DelAll(&LB)

First(LB) <- Nil

endif

} while (P!= Nil)

return LB

End

Procedure CopyList(Input L1:List, Input/Output L2:List)

// men-copy satu list ke list lain

Begin

L2 <- L1

End

List FCopyList(List L)

{ /\* Mengirimkan list yang merupakan salinan L \*/

}

void CpAlokList(List Lin, List \*Lout)

{ /\* I.S : Lin sembarang \*/

/\* F.S : Jika semua alokasi berhasil, maka Lout berisi \*/

/\* hasil copy Lin. Jika ada alokasi yang gagal \*/

/\* maka Lout=Nil, dan semua elemen yang terlanjur \*/

/\* dialokasi, didealokasi dengan melakukan alokasi \*/

/\* elemen. Lout adalah List kosong, jika ada alokasi\*/

/\* elemen yang gagal \*/

/\* Kamus \*/

address P,Pout

infotype X

/\* ALgoritma \*/

if(!ListEmpty(Lin)) { /\* tidak kosong \*/

CreateList(&(\*Lout))

P=First(Lin)

do

{ X=Info(P)

Pout=Alokasi(X)

if(Pout!=Nil) { /\* Alokasi berhasil \*/

InsertLast(&(\*Lout),Pout)

P=Next(P)

} else { /\* Alokasi Pout gagal \*/

DelAll(&(\*Lout))

First(\*Lout)=Nil

break

}

} while (Next(P) != Nil)

}

}

Procedure konkat(Input L1:List, Input L2:List, Input/Output L3:List)

// Menggabungkan List 1 dengan List 2 pada List 3

Begin

P1,P2,P3 : address

X1,X2 : infotype

CreateList(&(L3))

P1 <- First(L1)

do

{

X1 <- Info(P1)

P3 <- Alokasi(X1)

if(P3!=Nil) then

InsertLast(&(L3),P3)

P1 <- Next(P1)

else

DelAll(&(L3))

First(L3) <- Nil

break

endif

} while(Next(P1) != Nil)

P2 <- First(L2)

do

{

X2 <- Info(P2)

P3 <- Alokasi(X2)

if(P3!=Nil) then

InsertLast(&(L3),P3)

P2 <- Next(P2)

else

DelAll(&(L3))

First(L3) <- Nil

break

endif

}while (Next(P2)!=Nil)

End

void konkat1(List \*L1, List \*L2, List \*L3)

{ /\* I.S : L1 dan L2 sembarang ; \*/

/\* F.S : L1 dan L2 kosong, L3 adalah hasil konkatenasi\*/

/\* L1 & L2, \*/

/\* Konkatenasi 2 buah list : L1 dan L2 menghasilkan L3 \*/

/\* yang baru (dengan elemen list L1 dan L2 menjadi \*/

/\* List kosong. Tidak ada alokasi/dealokasi \*/

address P1,P2,P3

infotype X1,X2,X3

CreateList(&(\*L3))

while (First(\*L1)!=Nil) /\* L1 belum kosong \*/

{ DelFirst(&(\*L1),&P1)

InsertLast(&(\*L3),P1)

} /\* First(L1) == Nil Kosong \*/

while (First(\*L2)!=Nil) /\* L2 belum kosong \*/

{ DelFirst(&(\*L2),&P2)

InsertLast(&(\*L3),P2)

} /\* First (L2) == Nil ; kosong \*/

}

Procedure PecahList(Input/Output L1:List, Input/Output L2:List, Input L:List)

// Membagi dua list dari List L ke L1 dan L2

Begin

P,P1,P2 : address

X1,X2 : infotype

Nb,tengah : int

if(!ListEmpty(L)) then

CreateList(&(L1))

if(NbElmt(L)==1) then

First(L1) <- First(L)

else

tengah <- NbElmt(L) / 2

P <- First(L)

Nb <- 1

do

{ X1 <- Info(P)

P1 <- Alokasi(X1)

if(P1!=Nil) then

InsertLast(&(\*L1),P1)

P <- Next(P)

Nb++

else

DelAll(&(\*L1))

First(\*L1)=Nil

break

endif

} while(Nb<=tengah)

do

{

X2 <- Info(P)

P2 <- Alokasi(X2)

if(P2!=Nil) then

InsertLast(&(L2),P2)

P <- Next(P)

else

DelAll(&(L2))

First(L2) <- Nil

break

endif

} while(Next(P)!=Nil)

endif

else

printf("List tidak dapat dipecah ! Kosong !\n")

endif

End