{ Pseudocode Algorithm of linked list }

record Elmntlist

<

info : char

next : pointer of Elmntlist

>

record List

<

First : pointer of Elmntlist

>

procedure CreateList(output L : List )

begin

return L.First <- Null

end procedure CreateList

function Alokasi(input X : char) -> pointer of ElmntList

begin

pointer of ElmntList P <- Alokasi(X)

P <- alloc(ElmntList)

if (P not Null)

then

P^.Info <- X

P^.Next <- null

endif

return P

end function Alokasi

procedure Dealokasi(input/output P : pointer of ElmntList)

begin

P.next <- null

free(P)

end procedure Dealokasi

procedure InsVFirst(input/output L : List, input X : char)

begin

pointer of ElmntList P <- Alokasi(X)

if ( P not Null)

then

InsertFirst(L,P)

endif

end procedure InsVFirst

procedure InsertFirst(input/output L : List, output P : pointer of ElmntList)

begin

P^.Next <- L.First

L.First <- P

end procedure InsertFirst

procedure InsertLast(input/output L : List, output P : pointer of ElmntList)

begin

pointer of ElmntList Last

if (ListEmpty(L))

then

InsertFirst(L,P)

else

Last <- L.First

while (Last.next not null)

then

Last <- Last.next

endwhile

Last.next <- P

endif

end procedure InsertFirst

function ListEmpty(input L : List) -> boolean

begin

return L.First = null

end function ListEmpty

procedure InsVLast(input/output L : List, input X : char)

begin

pointer of ElmntList P <- Alokasi(X)

if (P not null)

then

InsertLast(L,P)

endif

end procedure InsVLast

procedure PrintInfo(input L : List)

begin

int i

pointer of ElmntList P <- L.First

if (P not null)

then

write("List Kosong !\n")

else

write("[ ")

do

write(P^.Info)

P <- P.next

i++

while P not null

write("]")

endif

write("\n")

end procedure PrintInfo

procedure DelP(input/output L : List, input X : char)

begin

pointer of ElmntList P <- L.First

pointer of ElmntList Pdel

while (P^.next not null and P.info not X)

then

P <- P.next

endwhile

if (P.info = X)

then

DelAfter(L,Pdel,P)

Dealokasi(Pdel)

endif

end procedure DelP

procedure DelAfter(input/output L : List, input/output Pdel : pointer of ElmntList, input/output Prec : pointer of ElmntList)

begin

Pdel <- Prec^.next

Prec^.next <- Pdel^.next

Pdel^.next <- null

end procedure DelAfter

procedure InsertVAfter(input/output L : List, input target : char, input x : char)

begin

pointer of ElmntList P <- Search1(L,target)

pointer of ElmntList Q <- Alokasi(x)

if (P not null)

then

Q^.next <- P^.next

P^.next <- Q

endif

end procedure InsertVAfter

function Search1(input L : List, input X : char) -> pointer of ElmntList

begin

pointer of ElmntList P <- L.First

boolean found <- false

while(P not null and nor found)

then

if (P.info = X)

then

found <- true

else

P <- P^.next

endif

endwhile

return P

end function Search1

{ Main Module }

begin

List L

pointer of ElmntList P

CreateList(L)

InsVFirst(L,'u')

PrintInfo(L)

InsVLast(L,'c')

PrintInfo(L)

InsVLast(L,'y')

PrintInfo(L)

InsVLast(L,'n')

PrintInfo(L)

InsVLast(L,'g')

PrintInfo(L)

InsVFirst(L,'k')

PrintInfo(L)

InsVLast(L,'h')

PrintInfo(L)

DelP(L,'c')

PrintInfo(L)

InsertVAfter(L,'c','i')

PrintInfo(L)

DelLast(address of L, address of P)

PrintInfo(L)

DelFirst(address of L, address of P)

PrintInfo(L)

end