20) Delete Value After

```
Procedure Dele_After_Value(* pHead, x: address, val: InfoType, * x : InfoType)
  If pHead is NULL Then
  End If

  Declare curr as address ← pHead

  While curr is not NULL and Info(curr) ≠ val Do
    curr ← Next(curr)
  End while

  If curr is not NULL and Next(curr) is Not NULL
    Declare temp as address ← Next(curr)
    x ← Info(temp)
    Next(curr) ← Next(temp)
    Deallocate temp
  End If
End procedure
```

2025.03.09 22:58

(8) Delete After

```
Procedure Del_After (*pBef : address, *x : integertype).
  If pBef is NULL or Next(pBef) is NULL Then

  Else
    Declare temp as address ← Next(pBef)
    x ← Info(temp)
    Next(pBef) ← Next(temp)
    Realocate temp
  End If
End Procedure
```

17) Delete Author

```
Procedure Del_Author(*p:address, *x:infotype):
  If p is NULL Then

  Else If Next(p) is NULL Then
      x ← info(p)
      Deallocate p
      p ← NULL
  Else
      Declare prev as address ← NULL
      Declare curr as address ← p

      While Next(curr) is not NULL DO
          prev ← curr
          curr ← Next(curr)
      Endwhile

      x ← info(curr)
      Next(prev) ← NULL
      Deallocate curr
  End If
End Procedure
```

No.

Date

16) Delete Awal

```
Procedure Del_Awal(* p:address, * x :infotype):
  IF p is NULL Then

  Else
     Declare temp as address ← p
     x ← Info(temp)
     p ← Next(temp)
     Deallocate temp
  End If
End Procedure
```

15) Modify ~~Set~~ Value Before

```
Procedure ModifyBeforeValue(*p: address, pAft: address,
                            *nval: infotype)
    If p is NULL or p is pAft Then

    End If

    Declare prev as address ← NULL
    Declare curr as address ← P

    While curr is not NULL and Next(curr) is not pAft:
        prev ← curr
        curr ← Next(curr)
    End while

    If curr is NULL Then
        nval ← -1
    Else
        Info(Prev) ← nval
    End if
End Procedure
```

2025.03.09 22:57

No.

Date

Fold

14) Modify Value After

```
Procedure ModifyAfterValue(*pRef: address, *nVal : integertype)
    If *pRef is not NULL and Next(pRef) is not NULL:
        Info(Next(pRef)) ← nVal
    End If
End procedure
```

No.

Date

Fold

11) Modify Akhir Address

```
Procedure Modify_Akhir_Address(*p, *Pnew: address)
  If p is NULL Then
    Pnew ← NULL
  Else
    Declare last as address ← p
    While Next(last) is not NULL Do
      last ← Next(last)
    End While

    Pnew ← last
  End If
End Procedure
```

No.

Date

Fold

10). Modify Awal Address

```
Procedure Modify_Awal_Address(* P:address, *Pnew:ada)
  If p is not NULL Then
     Pnew ← P
     P ← Next(P)
     Next(Pnew) ← NULL
  End If
End Procedure
```

2025.03.09 22:57

8) Insert value / after

```
Procedure InsertAfterValue(pBef: address, nilai: infotype)
    If pBef is NULL Then

    Else
        Declare Pnew as address ← createNode(nilai)
        Next(Pnew) ← Next(pBef)
        Next(pBef) ← Pnew
    End If
End procedure
```

No.

Date

6) Insert between/after

```
Procedure InsertAfter (* pBef: address, Pnew: address);
  If *pBef is NULL Then
     *pBef ← Pnew
  Else
     Next(Pnew) ← Next(pBef)
     Next(pBef) ← Pnew
  End If
End Procedure
```

5). ~~Ins~~ Insert after

```
Procedure Ins_After(*p : address, Pnew : address).
  If *p is NULL Then
    *p ← Pnew
  Else
    Next(Pnew) ← NULL
    Declare last as address ← *p

    While Next(last) is not NULL:
      last ← Next(last)
    End while

    Next(last) ← Pnew

  End If
End Procedure
```

2025.03.09 22:56

4). Insert Awal

```
Procedure Ins_Awal (*p: address, Pnew: address):
    If p is NULL Then
        p <- Pnew
    Else
        Next(Pnew) <- p
        p <- Pnew
    End If
End Procedure
```

3) Insert Node

```
Procedure Isi_Node (*p: address, x: infotype):
    Info(p) ← x
End procedure

Procedure insertNode (*p: address, nilai: infotype):
    If P is NULL Then:
        Create_Node (p)
        Info(p) ← nilai
        Next(p) ← NULL
    Else
        Declare q as address
        Create_Node(q)
        Info(q) ← nilai
        Next(q) ← P
        P ← q
    End if
End procedure
```

No.

Date

2) CreateNode (z)

Function createNode (x : infotype) → address
Alloc memory for p with size of ElmtList
Info(p) ← x
Next(p) ← NULL
Return p
End function

No.

Date

Fold

2). Create node (1)

```
Procedure Create_Node (*P : by ref address):
    Alloc memory for P with size of Elm + List
    Next (P) ← NULL

    If P is not NULL Then:
        Print
        Print
    End If
End procedure
```

No. Fauzi Ismail
241524042
Date

Fold

Define Info(P) as P.info
Define Next(P) as P.next

Type infotype as integer
Type address as Pointer to ElmtList

Structure ElmtList:
    info : infotype
    next : address
    prev : address

1). isEmpty

Function isEmpty(P: address) → boolean:
    If P is NULL Then:
        Return True
    Else
        Return false
    End If
End Function