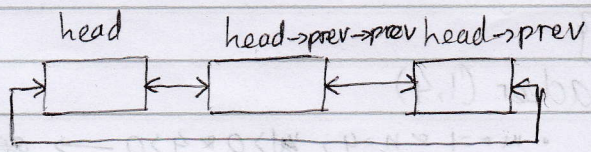
Struktur data

1. TNode * BeforeLast (TNode * head)

Ilustrasi.




```
{
    TNode* Bantu ;
    Bantu = head;
    if (Bantu -> next == head) // Satu Elemen.
    {
        return head ;
    }
    else // lebih dari 1 elemen.
    {
        return head -> prev -> prev ;
    }
}
```

2. Void InsertDescending (TNode * head, int databaru)

```
{
    TNode* Bantu, Baru
    Bantu = head ;
    Baru = new TNode* ;
    Baru -> data = databaru ;
    Baru -> next = NULL;
    if (Baru -> data < head -> data)
    {
        Baru -> next = head ;
        head = Baru ;
    }
    else if (Baru -> data ≤ tail -> data)
    {
        tail -> next = Baru ;
        tail = Baru ;
    }
    else
    {
        while (Baru -> data < Bantu -> data and Bantu -> next != tail)
        {
            Bantu = Bantu -> next ;
        }
```

```
        Baru -> next = Bantu -> next;
        Bantu -> next = Baru;
    }
}
```

3. acker (1,4)
   - m=1 & n=4; 1>0 & 4>0 → ~~acker (1,3)~~ acker (0, acker (1,3));

   acker (1,3)
   - m=1 & n=3; 1>0 & 3>0 → ~~acker (1,2)~~ acker (0, acker (1,2))

   acker (1,2)
   - m=1 & n=2; 1>0 & 2>0 → acker (0, acker (1,1))

   acker (1,1)
   - m=1 & n=1; 1>0 & 1>0 → acker (0, acker (1,0))

   acker (1,0)
   - m=1 & n=0; 1>0 & 0=0 → acker (0,1)

   acker (0,1)
   - m=0; 0=0 → n+1 = 2. → acker (1,0) = 2.

   - acker (0, acker (1,0)); acker (1,0) = 2.
   - acker (0, 2)
     - m=0, n=2; 0=0 → n+1 = 3. → acker (1,1) = 3

   - acker (0, acker (1,1)); acker (1,1) = 3
   - acker (0,3)
     - m=0, n=3; 0=0 → n+1 = 4. → acker (1,2) = 4

   - acker (0, acker (1,2)); acker (1,2) = 4
   - acker (0,4)
     - m=0, n=4; 0=0 → n+1 = 5 → acker (1,3) = 5

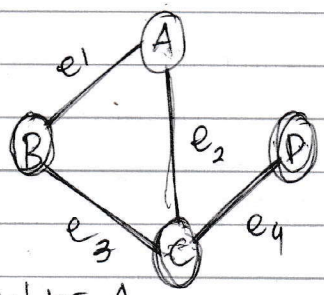
   - acker (0, acker (1,3)); acker (1,3) = 5
   - acker (0,5)
     - m=0, n=5; 0=0 → n+1 = 6 → acker (1,4) = 6.

∴ nilai dari acker (1,4) adalah 6 //

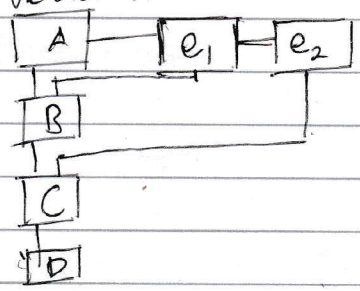
4. tree* HanyaKanan (tree* Pohon)

```
{
    if (isEmpty = True)
    {
        return NULL
    }
    else if (pohon->right = NULL && pohon->left = NULL)
    {
        return NULL
    }
    else if ( tobe Continued).
```

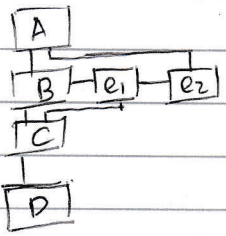
5.



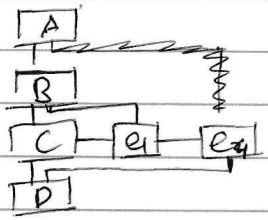
Vektor A



Vector B



Vector C



Vector D