

MODUL DAN GUIDED
PRAKTIKUM INFORMASI DAN STRUKTUR DATA

SEMESTER GANJIL 2022/2023

MODUL 7
Linked List 2



PROGRAM STUDI INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ATMA JAYA YOGYAKARTA
2022

Daftar Isi

I. Tujuan.....2

II. Dasar Teori.....2

Operasi pada linked list.....2

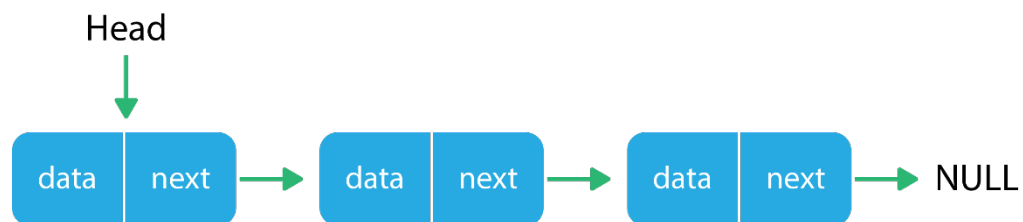
- Insert First3
- Insert After3
- Insert Last.....4
- Delete First.....5
- Delete At.....6
- Delete Last7

III. Guided.....8

I. Tujuan

1. Praktikan dapat semakin menguasai konsep Linked List Linear
2. Praktikan dapat mengembangkan konsep Linked List Linear

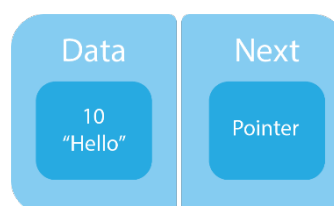
II. Dasar Teori



Gambar 2.1

Jika dilihat dari ilustrasi di atas, maka terlihat seakan-akan elemen dari list tersebut tersusun dan memiliki alamat yang berdampingan. Akan tetapi, sebenarnya elemen-elemen tersebut dialokasikan atau disimpan secara dynamic menggunakan malloc sehingga list-list tersebut akan terpecah-pecah didalam memori. Agar suatu elemen tersebut bisa mengakses elemen selanjutnya, maka kita akan membutuhkan pointer untuk menunjukkan alamat dari elemen yang akan diakses selanjutnya sehingga semua elemen bisa terhubung dan membentuk list seperti ilustrasi diatas.

Sebuah elemen terdiri dari dua bagian, yaitu data dan next (Pointer Next). Bagian data digunakan untuk menyimpan data yang akan ditampilkan atau digunakan, sedangkan bagian **pointer next** digunakan untuk menyimpan alamat dari node selanjutnya yang mengikutinya. Jika node tersebut sudah terletak pada bagian terakhir atau tidak ada node yang mengikutinya lagi, maka pointer next akan menunjuk **NULL**. Dengan cara ini maka kita bisa memanfaatkan memori dengan lebih efisien karena penggunaan memori dengan linked list dapat disesuaikan dengan mudah sesuai kebutuhan



Gambar 1.1

Operasi pada linked list

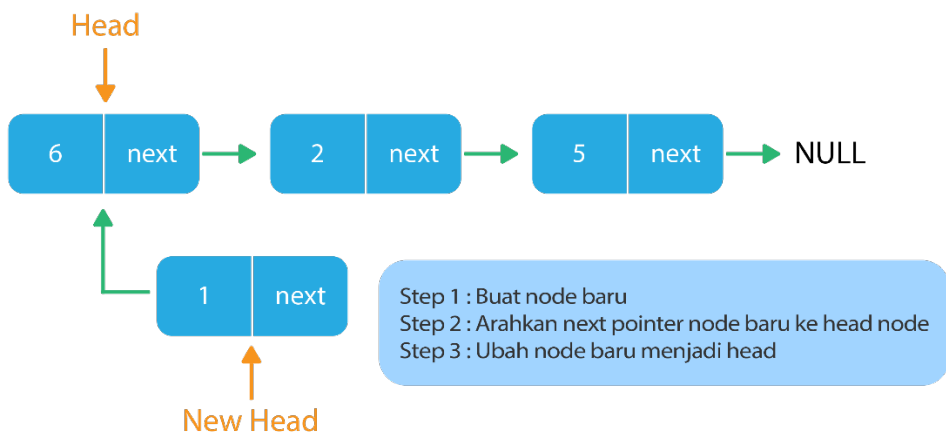
Operasi yang terpenting pada linked list yaitu menambahkan node (*insert*) dan menghapus node (*delete*)

Fungsi **Insert** pada linked list meliputi:

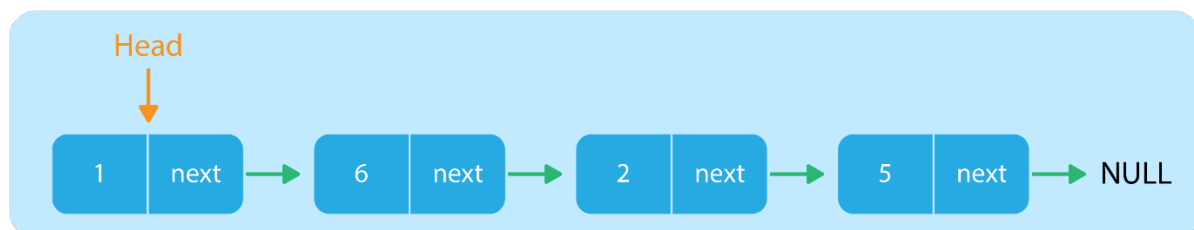
- Insert First
- Insert After
- Insert Last

- **Insert First**

(Insert sebagai node awal dari linked list)



Result :



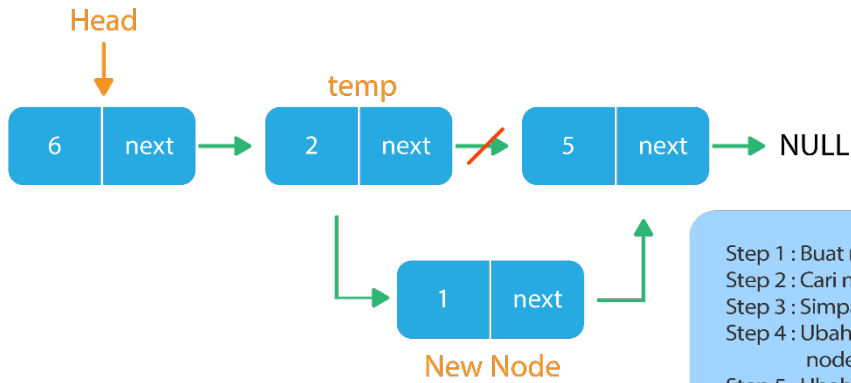
```

void insertFirst(List *L, infotype D)
{
    address P;

    P = alokasiData(D);
    P->next = L->first;
    L->first = P;
}

```

- **Insert After**
(Insert setelah node tertentu)



Step 1 : Buat node baru
 Step 2 : Cari node yang akan diinput setelahnya
 Step 3 : Simpan node tersebut ke variabel temp
 Step 4 : Ubah alamat pointer next node baru ke node setelah temp
 Step 5 : Ubah alamat pointer next variabel temp ke new node

Result :

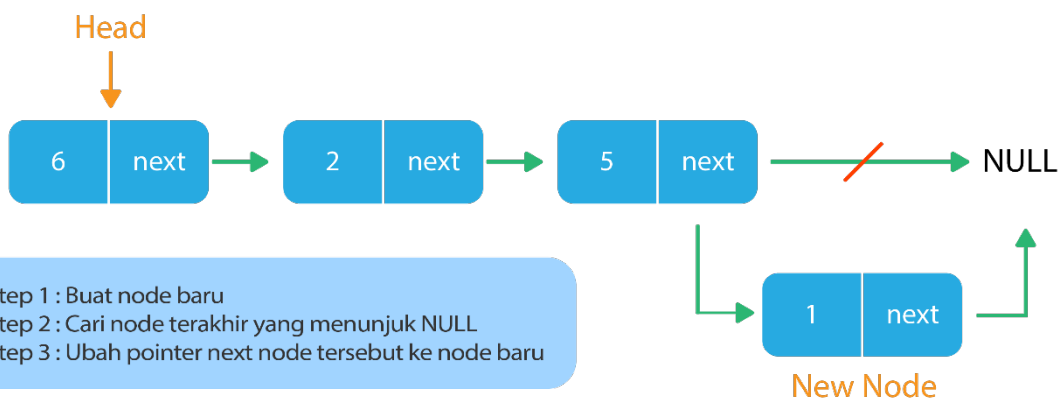


```
void insertAfter(List *L, infotype D, infotype previous)
{
    address P, before;
    before = findNode((*L), previous);

    if(before != NULL)
    {
        P = alokasiData(D);
        P->next = before->next; before->next = P;
    }
}

address findNode(List L, infotype X)
{
    address P;
    for(P = L.first; P != NULL && P->D != X; P = P->next);
    return P;
}
```

- **Insert Last**
(Insert di akhir dari linked list)



Result :



```

void insertLast(List *L, infotype D)
{
    address P, last;

    if(isEmpty(*L))
    {
        insertFirst(&(*L), D);
    }
    else
    {
        P = alokasiData(D);
        last = L->first;

        while(last->next != NULL)
        {
            last = last->next;
        }

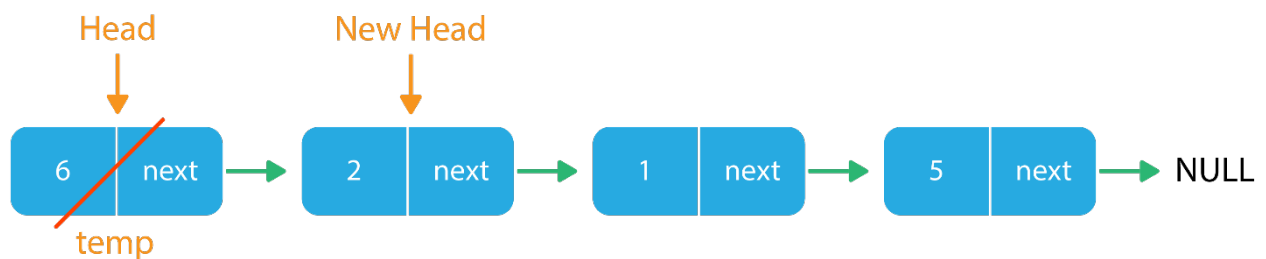
        last->next = P;
    }
}

```

Fungsi **Delete** pada linked list meliputi:

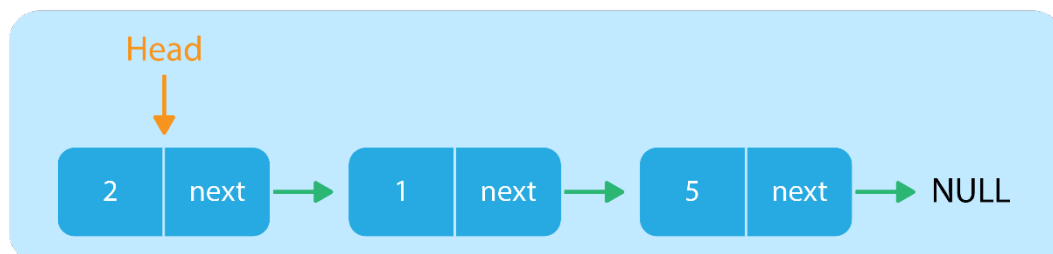
- Delete First
- Delete At
- Delete Last

- **Delete First**
(Penghapusan node pertama)



Step 1 : Simpan node head ke variabel temp
 Step 2 : Ubah node pointer next dari head menjadi head
 Step 3 : Ubah node baru menjadi head
 Step 4 : Hapus node temp

Result :

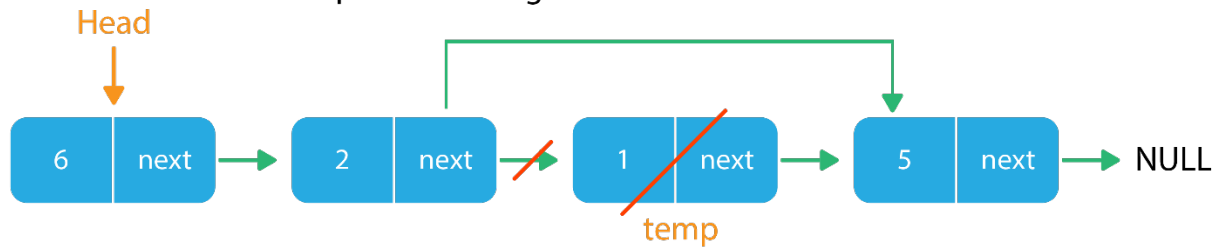


```
void deleteFirst(List *L)
{
    address del;

    if(!isEmpty((*L)))
    {
        del = L->first;
        L->first = L->first->next;
        free(del);
    }
}
```

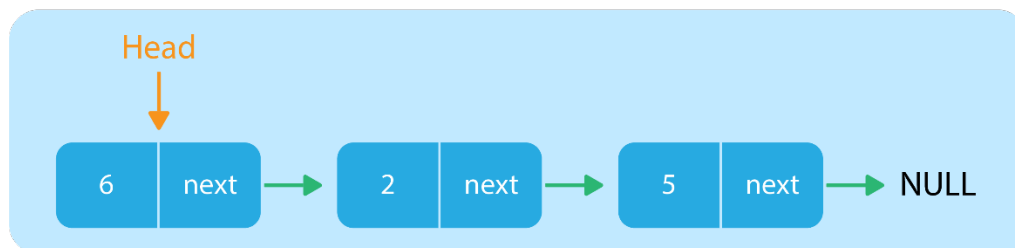
- **Delete At**
(Penghapusan pada spesifik node)

Hapus Node dengan data = 1



Step 1 : Cari node yang pointer next-nya menunjuk node yang akan dihapus
 Step 2 : Masukkan node yang ditunjuk pointer next ke variabel temp
 Step 3 : Hapus node temp
 Step 4 : Ubah alamat pointer next ke alamat node dari temp next

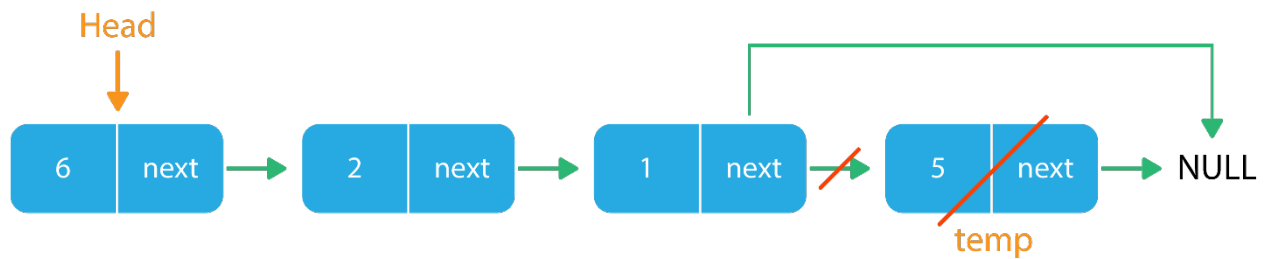
Result :



```
void deleteAt(List *L, infotype data)
{
    address P, del;
    del = findNode((*L), data);

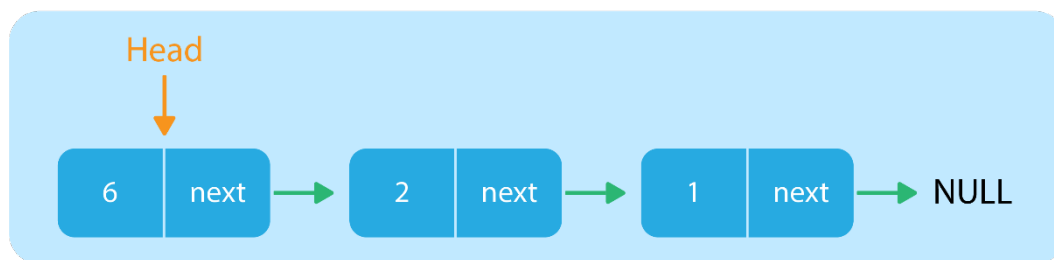
    if(del != NULL)
    {
        if(L->first == del)
        {
            deleteFirst(L);
        }
        else
        {
            for(P = (*L).first; P->next!=del; P = P->next);
            P->next = del->next;
            free(del);
        }
    }
}
```


- **Delete Last**
(Penghapusan node terakhir)



Step 1 : Cari node yang next keduanya menunjuk NULL
 Step 2 : Masukkan node terakhir ke variabel temp
 Step 3 : Hapus node temp
 Step 4 : Ubah pointer next node yang ditemukan tadi menjadi NULL

Result :



```
void deleteLast(List *L)
{
    address P;

    if(!isEmpty((*L)))
    {
        if(isOneElement((*L)))
        {
            deleteFirst(L);
        }
        else
        {
            for(P = L->first; P->next->next != NULL; P = P->next);
            free(P->next);
            P->next = NULL;
        }
    }
}
```

III. Guided

header.h

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <stdbool.h>
4 #include <conio.h>
5
6 typedef int infotype;           // Tipe data bentukan
7 typedef struct node *address;  // Tipe data bentukan untuk menyimpan address dari node
8
9 typedef struct node
10 {
11     infotype data;             // Data yang disimpan dalam node
12     address next;              // Address yang menunjuk node berikutnya
13 }node;                         // Tipe data node yang berisi data dan Next Pointer
14
15 typedef struct
16 {
17     address first;             // Berisi alamat node pertama dalam list
18 }List;
19
20 void createEmpty(List *L);      // Membuat List kosong
21 bool isEmpty(List L);           // Mengecek apakah sebuah List kosong
22 bool isOneElement(List L);      // Mengecek apakah List hanya berisi satu Node
23 address alokasi(infotype X);    // Membentuk Node baru
24 void insertFirst(List *L,address newNode); // Memasukkan Node kebagian depan List
25 void insertAfter(address before,address newNode); // Memasukkan Node baru setelah node yang telah ditentukan
26 void insertLast(List *L,address newNode); // Memasukkan Node pada bagian belakang List
27 void deleteFirst(List *L);      // Menghapus Node pada bagian depan
28 void deleteAt(List *L,address del); // Menghapus Node yang telah ditentukan
29 void deleteLast(List *L);       // Menghapus Node pada bagian akhir List
30 void printData(List L);         // Menampilkan semua data pada Node yang terdapat dalam list
31 address findNode(List L,infotype X); // Mencari Node
32

```

source.c

```

1 #include "header.h"
2
3 void createEmpty(List *L)
4 {
5     (*L).first = NULL;
6 }
7
8 bool isEmpty(List L)
9 {
10     return L.first == NULL;
11 }
12

```

```

13 bool isOneElement(List L)
14 {
15     return !isEmpty(L) && L.first->next == NULL;
16 }
17
18 address alokasi(infotype X)
19 {
20     address P;
21     P = (node*) malloc(sizeof(node)); P->data = X;
22     P->next = NULL; return P;
23 }
24
25 void insertFirst(List *L, address newNode)
26 {
27     newNode->next = (*L).first; (*L).first = newNode;
28 }
29
30 void insertAfter(address before, address newNode)
31 {
32     if(before != NULL)
33     {
34         newNode->next = before->next; before->next = newNode;
35     }
36 }
37
38 void insertLast(List *L, address newNode)
39 {
40     address P;
41
42     if(isEmpty(*L))
43     {
44         insertFirst(L, newNode);
45     }
46     else
47     {
48         for(P = (*L).first; P->next != NULL; P = P->next);
49         P->next = newNode;
50     }
51 }
52
53 void deleteFirst(List *L)
54 {
55     if(!isEmpty(*L))
56     {
57         address del = (*L).first;
58         (*L).first = (*L).first->next;
59         free(del);
60     }
61 }
62

```

```

63 void deleteAt(List *L,address del)
64 {
65     address P;
66
67     if(!isEmpty((*L)))
68     {
69         if((*L).first == del)
70         {
71             deleteFirst(L);
72         }
73         else
74         {
75             P = (*L).first;
76             while(P->next != del)
77             {
78                 P = P->next;
79             }
80
81             P->next = del->next;
82             free(del);
83         }
84     }
85 }
86
87 void deleteLast(List *L)
88 {
89     address P;
90
91     if(!isEmpty((*L)))
92     {
93         if(isOneElement(*L))
94         {
95             deleteFirst(L);
96         }
97         else
98         {
99             for(P = (*L).first; P->next->next != NULL; P = P->next);
100             free(P->next);
101             P->next = NULL;
102         }
103     }
104 }
105
106 void printData(List L)
107 {
108     address P;
109
110     printf("\n\t");
111     for(P = L.first; P != NULL; P = P->next)
112         printf("%d ",P->data);
113 }
114
115 address findNode(List L,infotype X)
116 {
117     address P;
118
119     for(P = L.first; P != NULL && P->data != X; P = P->next);
120     return P;
121 }
122

```

main.c

```

1 #include "header.h"
2
3 int main() {
4     List L;
5     address temp,before;
6     infotype bil;
7     char menu;
8
9
10    createEmpty(&L);        // Inisialisasi List
11
12    do
13    {
14        system("cls");
15        printf("\n\n\t GUIDED LINKED LIST 2\n");
16        printf("\n\t 1. Insert First");
17        printf("\n\t 2. Insert After");
18        printf("\n\t 3. Insert Last");
19        printf("\n\t 4. Delete First");
20        printf("\n\t 5. Delete At");
21        printf("\n\t 6. Delete Last");
22        printf("\n\t 7. Print Data");
23        printf("\n\t 0. EXIT");
24        printf("\n\t Input Menu\t: "); menu = getch();
25        printf("\n\t\n");
26
27        switch(menu)
28        {
29            case '1':
30                printf("\n\t Enter Number\t: "); scanf("%d",&bil);
31                temp = alokasi(bil);
32                insertFirst(&L,temp);
33                printf("\n\t Inserted Successfully");
34                break;
35
36            case '2':
37                printf("\n\t Insert number after : ");
38                scanf("%d",&bil);
39
40                before = findNode(L,bil);
41                if(before==NULL)
42                {
43                    printf("\n\t [!] 404 Number not found");
44                    break;
45                }
46
47                printf("\n\t Enter Number\t: "); scanf("%d",&bil);
48                temp = alokasi(bil);
49                insertAfter(before,temp);
50                printf("\n\t Inserted Successfully");
51                break;
52
53            case '3':
54                printf("\n\t Enter number\t: "); scanf("%d",&bil);
55                temp = alokasi(bil);
56                insertLast(&L,temp);
57                printf("\n\t Inserted Successfully");
58                break;
59

```

```

60         case '4':
61             if(isEmpty(L))
62             {
63                 printf("\n\t [!] List kosong!");
64                 break;
65             }
66             deleteFirst(&L);
67             printf("\n\t Delete successfully");
68             break;
69
70         case '5':
71             if(isEmpty(L))
72             {
73                 printf("\n\t [!] List is Empty!");
74                 break;
75             }
76
77             printf("\n\tNumber you want to delete\t:"); scanf("%d",&bil);
78             temp = findNode(L,bil);
79
80             if(temp==NULL)
81             {
82                 printf("\n\t [!] 404 Number not found");
83                 break;
84             }
85
86             deleteAt(&L,temp); printf("\n\t Deleted Successfully");
87             break;
88
89         case '6':
90             if(isEmpty(L))
91             {
92                 printf("\n\t [!] List is Empty!");
93                 break;
94             }
95
96             deleteLast(&L);
97             printf("\n\t Deleted Successfully");
98             break;
99
100        case '7':
101            if(isEmpty(L))
102            {
103                printf("\n\t [!] List is Empty!");
104                break;
105            }
106
107            printData(L);
108            break;
109
110        case '0':
111            printf("\n\t Exit");
112            break;
113
114        default :
115            printf("\n\t[!] Menu doesn't exist");
116            break;
117    }
118    getch();
119 }while(menu!='0');
120
121 return 0;
122 }

```

Untuk keperluan UGD (Optional):

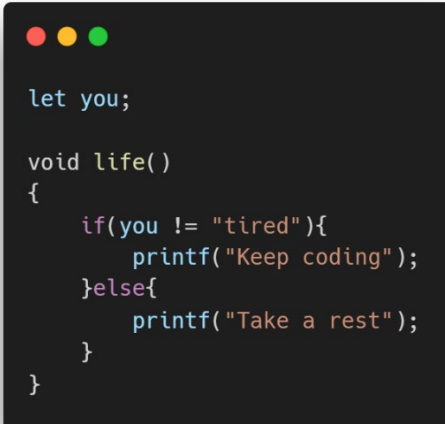
Pelajari penerapan sorting pada linked list.

Format Pengumpulan:

GD7_X_YYYYY

X = Kelas

Y = 5 Digit NPM Terakhir



```
let you;

void life()
{
    if(you != "tired"){
        printf("Keep coding");
    }else{
        printf("Take a rest");
    }
}
```