1 D4 - TEKKOM B

PRAKTIKUM 4

SINGLY LINKED LIST

Nama	:	Septian Bagus Jumantoro
Kelas	:	1 – D4 Teknik Komputer B
NRP	:	3221600039
Dosen	:	Dr Bima Sena Bayu Dewantara S.ST, MT.
Mata Kuliah	:	Praktikum Pemrograman Dasar 2
Hari/Tgl. Praktikum	:	Rabu, 30 Maret 2022

➢ **1a.c**

```c
#include <stdio.h>
#include <stdlib.h>
struct Node
{
    int data;
    struct Node* next;
};
int main(){
    struct Node* head = NULL;
    struct Node* second = NULL;
    struct Node* third = NULL;
    struct Node* fourth = NULL;
    struct Node* fifth = NULL;
    struct Node* sixth = NULL;
    struct Node* seventh = NULL;

    head = (struct Node*)malloc(sizeof(struct Node));
    second = (struct Node*)malloc(sizeof(struct Node));
    third = (struct Node*)malloc(sizeof(struct Node));
    fourth = (struct Node*)malloc(sizeof(struct Node));
    fifth = (struct Node*)malloc(sizeof(struct Node));
    sixth = (struct Node*)malloc(sizeof(struct Node));
    seventh = (struct Node*)malloc(sizeof(struct Node));

    head->data = 1;
    head->next = second;

    second->data = 2;
    second->next = third;

    third->data = 3;
    third->next = fourth;

    fourth->data = 4;
    fourth->next = fifth;

    fifth->data = 5;
    fifth->next = sixth;

    sixth->data = 6;
    sixth->next = seventh;

    seventh->data = 7;
    seventh->next = NULL;

    return 0;
}
```

➢ **1b.cpp**

```cpp
#include <bits/stdc++.h>
using namespace std;

class Node
{
    public:
            int data;
            Node* next;
};

int main(){
    Node* head = NULL;
    Node* second = NULL;
    Node* third = NULL;
    Node* fourth = NULL;
    Node* fifth = NULL;
    Node* sixth = NULL;
    Node* seventh = NULL;

    head = new Node();
    second = new Node();
    third = new Node();
    fourth = new Node();
    fifth = new Node();
    sixth = new Node();
    seventh = new Node();

    head->data = 1;
    head->next = second;

    second->data = 2;
    second->next = third;

    third->data = 3;
    third->next = fourth;

    fourth->data = 4;
    fourth->next = fifth;

    fifth->data = 5;
    fifth->next = sixth;

    sixth->data = 6;
    sixth->next = seventh;

    seventh->data = 7;
    seventh->next = NULL;
    return 0;
}
```

- **Tugas**

  Perbedaannya:

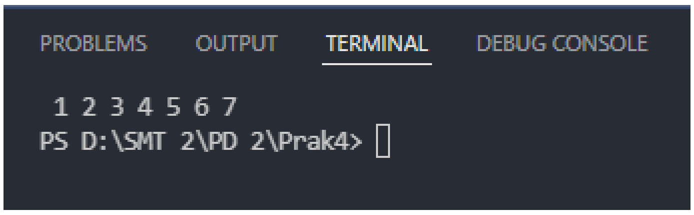
  Pada 2a.c menggunakan bahasa C dan menggunakan Struct dan malloc

  Pada 2b.cpp menggunakan bahasan C++ dan menggunakan Class dan new

- **2a.c**

```c
#include <stdio.h>
#include <stdlib.h>

struct Node
{
    int data;
    struct Node* next;
};

void printList(struct Node* n){
    while (n != NULL){
        printf(" %d", n->data);
        n = n->next;
    }
}

int main(){
    system("cls");
    struct Node* head = NULL;
    struct Node* second = NULL;
    struct Node* third = NULL;
    struct Node* fourth = NULL;
    struct Node* fifth = NULL;
    struct Node* sixth = NULL;
    struct Node* seventh = NULL;

    head = (struct Node*)malloc(sizeof(struct Node));
    second = (struct Node*)malloc(sizeof(struct Node));
    third = (struct Node*)malloc(sizeof(struct Node));
    fourth = (struct Node*)malloc(sizeof(struct Node));
    fifth = (struct Node*)malloc(sizeof(struct Node));
    sixth = (struct Node*)malloc(sizeof(struct Node));
    seventh = (struct Node*)malloc(sizeof(struct Node));
    head->data = 1;
    head->next = second;

    second->data = 2;
    second->next = third;

    third->data = 3;
    third->next = fourth;

    fourth->data = 4;
    fourth->next = fifth;

    fifth->data = 5;
    fifth->next = sixth;

    sixth->data = 6;
    sixth->next = seventh;

    seventh->data = 7;
    seventh->next = NULL;

    printList(head);

    return 0;
}
```

➢ **Output**



```
 1 2 3 4 5 6 7
PS D:\SMT 2\PD 2\Prak4>
```

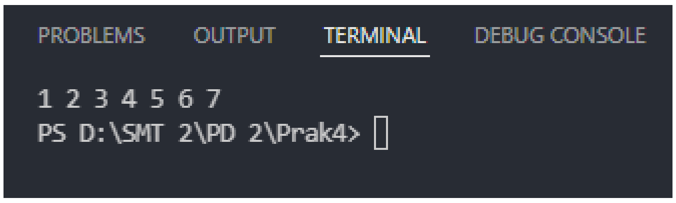
➢ **2b.cpp**

```
#include <bits/stdc++.h>
using namespace std;

class Node
{
    public:
    int data;
    Node* next;
};
```

```cpp
void printList(Node* n){
    while(n != NULL){
        cout << n->data << " ";
        //cout << &n->data;
        n = n->next;
    }
}

int main(){
   system("cls");
    Node* head = NULL;
    Node* second = NULL;
    Node* third = NULL;
    Node* fourth = NULL;
    Node* fifth = NULL;
    Node* sixth = NULL;
    Node* seventh = NULL;

    head = new Node();
    second = new Node();
    third = new Node();
    fourth = new Node();
    fifth = new Node();
    sixth = new Node();
    seventh = new Node();

    head->data = 1;
    head->next = second;

    second->data = 2;
    second->next = third;

    third->data = 3;
    third->next = fourth;

    fourth->data = 4;
    fourth->next = fifth;

    fifth->data = 5;
    fifth->next = sixth;

    sixth->data = 6;
    sixth->next = seventh;

    seventh->data = 7;
    seventh->next = NULL;

    printList(head);
```

```c
    return 0;
}
```

- **Output**

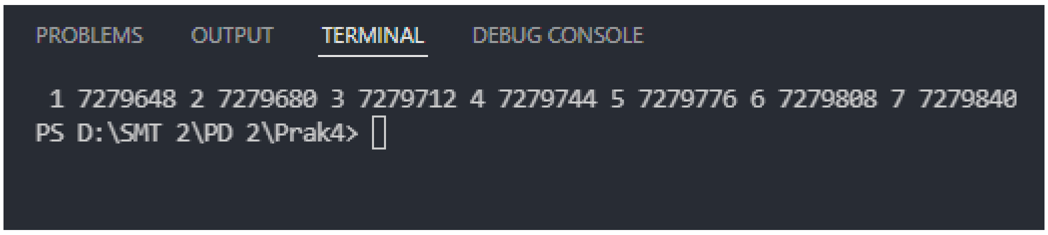


```
1 2 3 4 5 6 7
PS D:\SMT 2\PD 2\Prak4> 
```

- **2c.c**

```c
#include <stdio.h>
#include <stdlib.h>

struct Node
{
    int data;
    struct Node* next;
};

void printList(struct Node* n){
    while (n != NULL){
        printf(" %d %d", n->data, &n->data);
        n = n->next;
    }
}

int main(){
    system("cls");
    struct Node* head = NULL;
    struct Node* second = NULL;
    struct Node* third = NULL;
    struct Node* fourth = NULL;
    struct Node* fifth = NULL;
    struct Node* sixth = NULL;
    struct Node* seventh = NULL;

    head = (struct Node*)malloc(sizeof(struct Node));
    second = (struct Node*)malloc(sizeof(struct Node));
    third = (struct Node*)malloc(sizeof(struct Node));
    fourth = (struct Node*)malloc(sizeof(struct Node));
    fifth = (struct Node*)malloc(sizeof(struct Node));
    sixth = (struct Node*)malloc(sizeof(struct Node));
    seventh = (struct Node*)malloc(sizeof(struct Node));
    head->data = 1;
    head->next = second;

    second->data = 2;
    second->next = third;

    third->data = 3;
    third->next = fourth;

    fourth->data = 4;
    fourth->next = fifth;

    fifth->data = 5;
    fifth->next = sixth;

    sixth->data = 6;
    sixth->next = seventh;

    seventh->data = 7;
    seventh->next = NULL;

    printList(head);

    return 0;
}
```

- **Output**



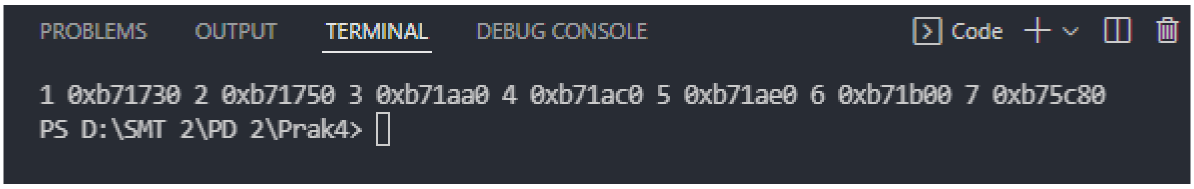
```
 1 7279648 2 7279680 3 7279712 4 7279744 5 7279776 6 7279808 7 7279840
PS D:\SMT 2\PD 2\Prak4> 
```

- **2c.cpp**

```
#include <bits/stdc++.h>
using namespace std;

class Node
{
    public:
    int data;
    Node* next;
};
```

```cpp
void printList(Node* n){
    while(n != NULL){
        cout << n->data << " ";
        cout << &n->data;
        n = n->next;
    }
}

int main(){
   system("cls");
    Node* head = NULL;
    Node* second = NULL;
    Node* third = NULL;
    Node* fourth = NULL;
    Node* fifth = NULL;
    Node* sixth = NULL;
    Node* seventh = NULL;

    head = new Node();
    second = new Node();
    third = new Node();
    fourth = new Node();
    fifth = new Node();
    sixth = new Node();
    seventh = new Node();

    head->data = 1;
    head->next = second;

    second->data = 2;
    second->next = third;

    third->data = 3;
    third->next = fourth;

    fourth->data = 4;
    fourth->next = fifth;

    fifth->data = 5;
    fifth->next = sixth;

    sixth->data = 6;
    sixth->next = seventh;

    seventh->data = 7;
    seventh->next = NULL;

    printList(head);
    return 0;
}
```

- **Output**



```
1 0xb71730 2 0xb71750 3 0xb71aa0 4 0xb71ac0 5 0xb71ae0 6 0xb71b00 7 0xb75c80
PS D:\SMT 2\PD 2\Prak4> 
```

- **Analisa**

Pada percobaan kedua ini merupakan lanjutan dari percobaan pertama, dimana pada percobaan kedua ini menampilkan output beserta alamat dari output tersebut.

Pada percobaan 2c.c untuk memanggil alamat dapat menggunakan syntax berikut:

```
printf(" %d %d", n->data, &n->data);
```

Pada percobaan 2c.cpp untuk memanggil alamat dapat menggunakan syntax berikut:

```
cout << n->data << " ";
        cout << &n->data;
```

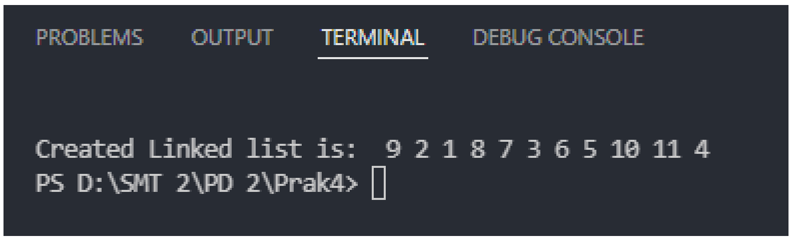
- **3a.c**

```
#include<stdio.h>
#include<stdlib.h>

struct Node{
    int data;
    struct Node *next;
};

void push(struct Node** head_ref, int new_data){
    struct Node* new_node =  (struct Node*)malloc(sizeof(struct Node));
    new_node->data = new_data;
    new_node->next = (*head_ref);
    (*head_ref) = new_node;
}

void insertAfter(struct Node* prev_node, int new_data){
    if (prev_node == NULL){
```

```c
        printf("the given previous node cannot be NULL");
        return;
    }
    struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));
    new_node->data = new_data;
    new_node->next = prev_node->next;
    prev_node->next = new_node;
}

void append(struct Node** head_ref, int new_data){
    struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));
    struct Node *last = *head_ref;
    new_node->data = new_data;
    new_node->next = NULL;

    if(*head_ref == NULL){
        *head_ref = new_node;
        return;
    }
    while(last->next != NULL)
        last = last->next;
        last->next = new_node;
        return;
}

void printList(struct Node *node){
    while(node != NULL){
        printf(" %d", node->data);
        node = node->next;
    }
}

int main(){
    system("cls");
    struct Node* head = NULL;
    append(&head,6);
    append(&head,5);
    insertAfter(head->next,10);
    push(&head,7);
    append(&head,11);
    push(&head,1);
    insertAfter(head->next,3);
    append(&head,4);
    push(&head,2);
    insertAfter(head->next,8);
    push(&head,9);

    printf("\nCreated Linked list is: ");
    printList(head);
```

```cpp
    return 0;
}
```

- **Output**

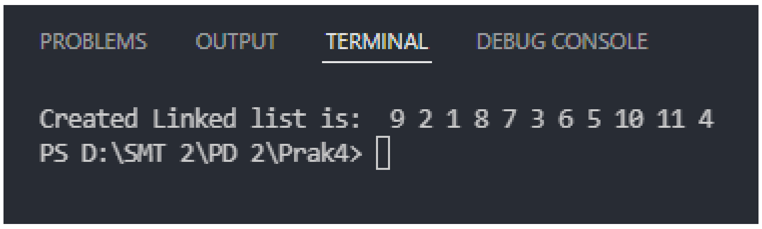


```
Created Linked list is:  9 2 1 8 7 3 6 5 10 11 4
PS D:\SMT 2\PD 2\Prak4> 
```

- **3b.cpp**

```cpp
#include<bits/stdc++.h>
using namespace std;

class Node{
    public:
    int data;
    Node *next;
};

void push(Node** head_ref, int new_data){
    Node* new_node = new Node();
    new_node->data = new_data;
    new_node->next = (*head_ref);
    (*head_ref) = new_node;
}

void insertAfter(Node* prev_node, int new_data){
    if (prev_node == NULL){
        cout << "the given previous node cannot be NULL";
        return;
    }
    Node* new_node = new Node();
    new_node->data = new_data;
    new_node->next = prev_node->next;
    prev_node->next = new_node;
}

void append(Node** head_ref, int new_data){
    Node* new_node = new Node();
    Node *last = *head_ref;
    new_node->data = new_data;
    new_node->next = NULL;
    if(*head_ref == NULL){
        *head_ref = new_node;
        return;
    }
    while(last->next != NULL)
        last = last->next;
        last->next = new_node;
        return;
}

void printList(Node *node){
    while(node != NULL){
        cout << " " << node->data;
        node = node->next;
    }
}

int main(){
    system("cls");
    Node* head = NULL;
    append(&head,6);
    append(&head,5);
    insertAfter(head->next,10);
    push(&head,7);
    append(&head,11);
    push(&head,1);
    insertAfter(head->next,3);
    append(&head,4);
    push(&head,2);
    insertAfter(head->next,8);
    push(&head,9);

    cout << "Created Linked list is: ";
    printList(head);

    return 0;
}
```

- **Output**



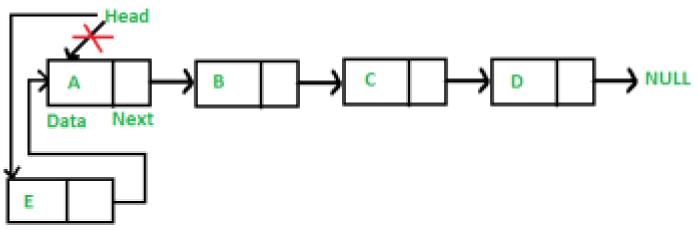
```
Created Linked list is:  9 2 1 8 7 3 6 5 10 11 4
PS D:\SMT 2\PD 2\Prak4> 
```

- **Analisa**

  Pada percobaan tersebut terdapat beberapa fungsi yang digunakan untuk memasukkan beberapa node.

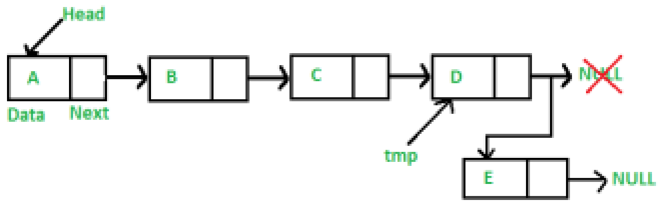
  Untuk fungsi Push digunakan untuk memasukkan data ke urutan pertama.

  Seperti:



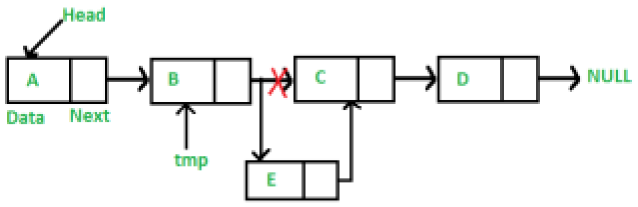
  Untuk fungsi Append digunakan untuk memasukkan data ke urutan paling belakang

  Seperti:



  Untuk fungsi insertAfter digunakan untuk memasukkan data setelah node yang telah ditentukan sebelumnya.

  Seperti:

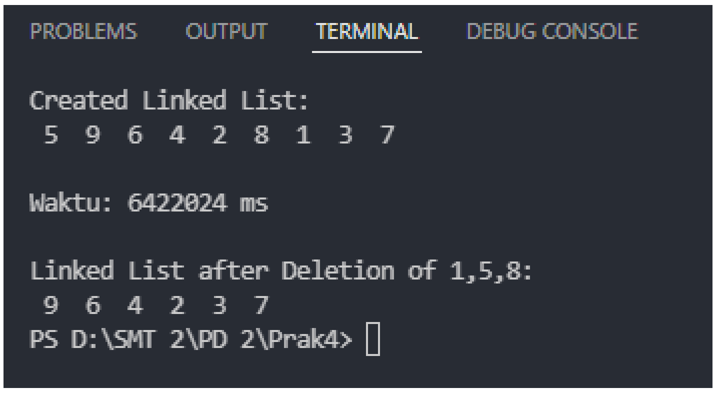


- **4a.c**

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

struct Node{
    int data;
    struct Node *next;
};

void push(struct Node** head_ref, int new_data){
    struct Node* new_node = (struct Node*) malloc(sizeof(struct Node));
    new_node->data = new_data;
    new_node->next = (*head_ref);
```

```c
    (*head_ref) = new_node;
}

void deleteNode(struct Node **head_ref, int key){
    struct Node* temp = *head_ref, *prev;
    if (temp != NULL && temp->data == key){
        *head_ref = temp->next;
        free(temp);
        return;
    }

    while (temp != NULL && temp->data != key){
        prev = temp;
        temp = temp->next;
    }

    if (temp == NULL)
    return;

    prev->next = temp->next;
    free(temp);
}

void printList(struct Node *node){
    while (node != NULL){
        printf(" %d ", node->data);
        node = node->next;
        }
}

int main(){
    system("cls");
    struct Node* head = NULL;

    push(&head, 7);
    push(&head, 1);
    push(&head->next, 3);
    push(&head, 2);
    push(&head->next, 8);
    push(&head, 4);
    push(&head, 9);
    push(&head->next, 6);
    push(&head, 5);

    puts("Created Linked List: ");
    printList(head);

    clock_t start;
    double duration;
    start = clock();

    deleteNode(&head, 1);
    deleteNode(&head, 5);
    deleteNode(&head, 8);
    duration = (clock() - start) / (double) CLOCKS_PER_SEC;
    puts(" ");
    printf("\nWaktu: %d ms", &duration);
    puts(" ");
    puts("\nLinked List after Deletion of 1,5,8: ");
    printList(head);
    return 0;
}
```

- **Output**



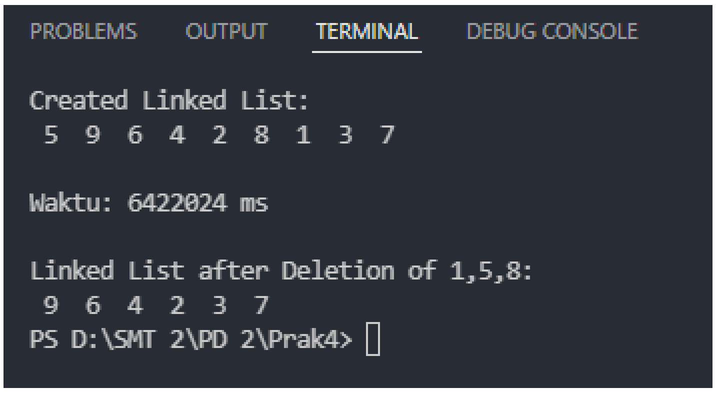
```
Created Linked List:
 5  9  6  4  2  8  1  3  7

Waktu: 6422024 ms

Linked List after Deletion of 1,5,8:
 9  6  4  2  3  7
PS D:\SMT 2\PD 2\Prak4>
```

- **4b.cpp**

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

struct Node{
    int data;
    struct Node *next;
};

void push(struct Node** head_ref, int new_data){
    struct Node* new_node = (struct Node*) malloc(sizeof(struct Node));
    new_node->data = new_data;
    new_node->next = (*head_ref);
    (*head_ref) = new_node;
}

void deleteNode(struct Node **head_ref, int position){
    struct Node* temp = *head_ref, *prev;
```

```c
    if (*head_ref == NULL){
        return;
    }
    struct Node* temp = *head_ref;
    if(position == 0){
        *head_ref = temp->next;
        free(temp);
    }
    for(int i=0; temp!=NULL && i<position-1; i++)
    temp = temp->next;

    if(temp == NULL || temp->next == NULL)
    return;

    struct Node *next = temp->next->next;
    free(temp->next);
    temp->next = next;

void printList(struct Node *node){
    while (node != NULL){
        printf(" %d ", node->data);
        node = node->next;
        }
}

int main(){
    system("cls");
    struct Node* head = NULL;

    push(&head, 7);
    push(&head, 1);
    push(&head, 3);
    push(&head, 2);
    push(&head, 8);
    push(&head, 4);
    push(&head, 9);
    push(&head, 6);
    push(&head, 5);

    puts("Created Linked List: ");
    printList(head);

    clock_t start;
    double duration;
    start = clock();

    deleteNode(&head, 1);
    deleteNode(&head, 5);
    deleteNode(&head, 8);
    duration = (clock() - start) / (double) CLOCKS_PER_SEC;
    puts(" ");
    printf("\nWaktu: %d ms", &duration);
    puts(" ");
    puts("\nLinked List after Deletion of 1,5,8: ");
    printList(head);
    return 0;
}
```

➢ **Output**



```
Created Linked List:
 5  9  6  4  2  8  1  3  7

Waktu: 6422024 ms

Linked List after Deletion of 1,5,8:
 9  6  4  2  3  7
PS D:\SMT 2\PD 2\Prak4> 
```

➢ **Analisa**

Pada percobaan 4a.c fungsi deleteNode berfungsi untuk menghapus nilai node yang dipilih dalam parameter. Apabila nilai yang dipilih tidak terdapat dalam linked list maka akan kembali, jika nilai tersebut tersedia maka nilai itu akan dihapus dan node dengan urutan selanjutnya akan menggantikan urutan node yang telah di delete (menggantikan tempatnya).

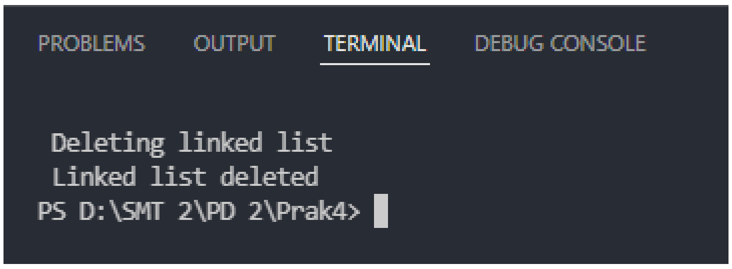
Pada percobaan 4b.c fungsi deleteNode berfungsi untuk menghapus nilai node sesuai dengan urutan dalam linked list. Apabila urutan yang dipilih kosong maka akan kembali, jika urutan yang dipilih lebih atau tidak ada dalam urutan, maka fungsi akan kembali. Jika urutan yang dipilih ada dalam urutan linked list maka nilai dari urutan tersebut akan di delete dan nilai pada urutan selanjutnya akan menggantikan posisi nilai yang telah didelete.

Pada kedua percobaan telah ditambahkan syntax waktu pada fungsi main untuk membaca waktu program berlangsung, dan pada hasil yang ditampilkan waktu tersebut bernilai sama.

- **5a.c**

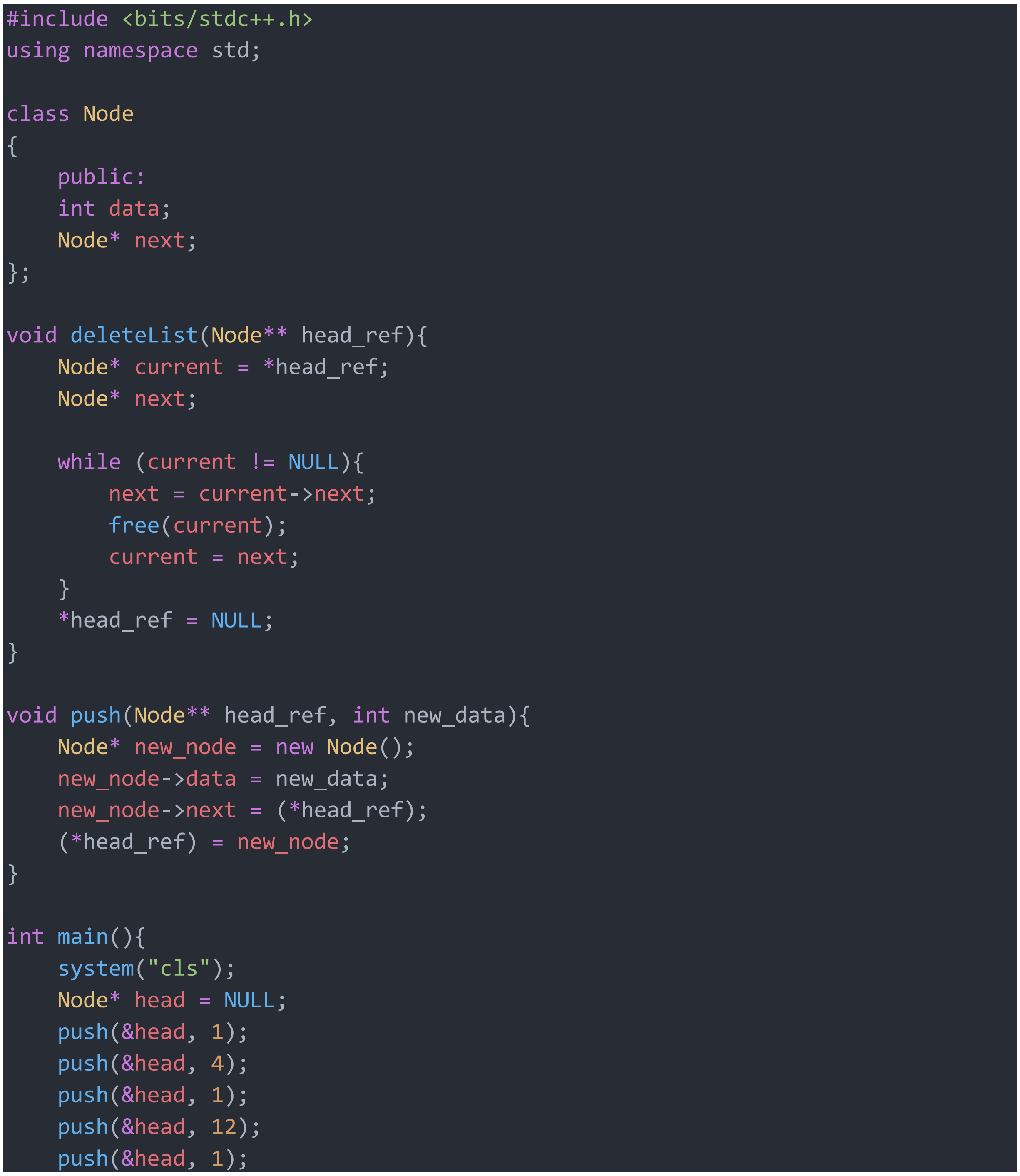
```c
#include <stdio.h>
#include <stdlib.h>
#include <assert.h>

struct Node
{
    int data;
    struct Node* next;
};

void deleteList(struct Node** head_ref){
    struct Node* current = *head_ref;
    struct Node* next;

    while (current != NULL){
        next = current->next;
        free(current);
        current = next;
    }
    *head_ref = NULL;
}

void push(struct Node** head_ref, int new_data){
    struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));
    new_node->data = new_data;
    new_node->next = (*head_ref);
    (*head_ref) = new_node;
}

int main(){
    system("cls");
    struct Node* head = NULL;
    push(&head, 1);
    push(&head, 4);
    push(&head, 1);
    push(&head, 12);
    push(&head, 1);

    printf("\n Deleting linked list");
    deleteList(&head);

    printf("\n Linked list deleted");

}
```

- **Output**



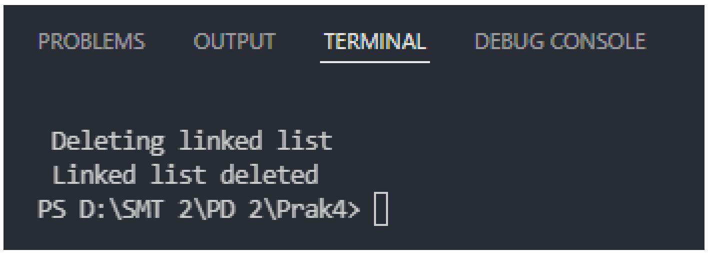
```
Deleting linked list
Linked list deleted
PS D:\SMT 2\PD 2\Prak4>
```

- **5b.cpp**



```cpp
#include <bits/stdc++.h>
using namespace std;

class Node
{
    public:
    int data;
    Node* next;
};

void deleteList(Node** head_ref){
    Node* current = *head_ref;
    Node* next;

    while (current != NULL){
        next = current->next;
        free(current);
        current = next;
    }
    *head_ref = NULL;
}

void push(Node** head_ref, int new_data){
    Node* new_node = new Node();
    new_node->data = new_data;
    new_node->next = (*head_ref);
    (*head_ref) = new_node;
}

int main(){
    system("cls");
    Node* head = NULL;
    push(&head, 1);
    push(&head, 4);
    push(&head, 1);
    push(&head, 12);
    push(&head, 1);
    cout << ("\n Deleting linked list");
    deleteList(&head);

    cout << ("\n Linked list deleted");

}
```

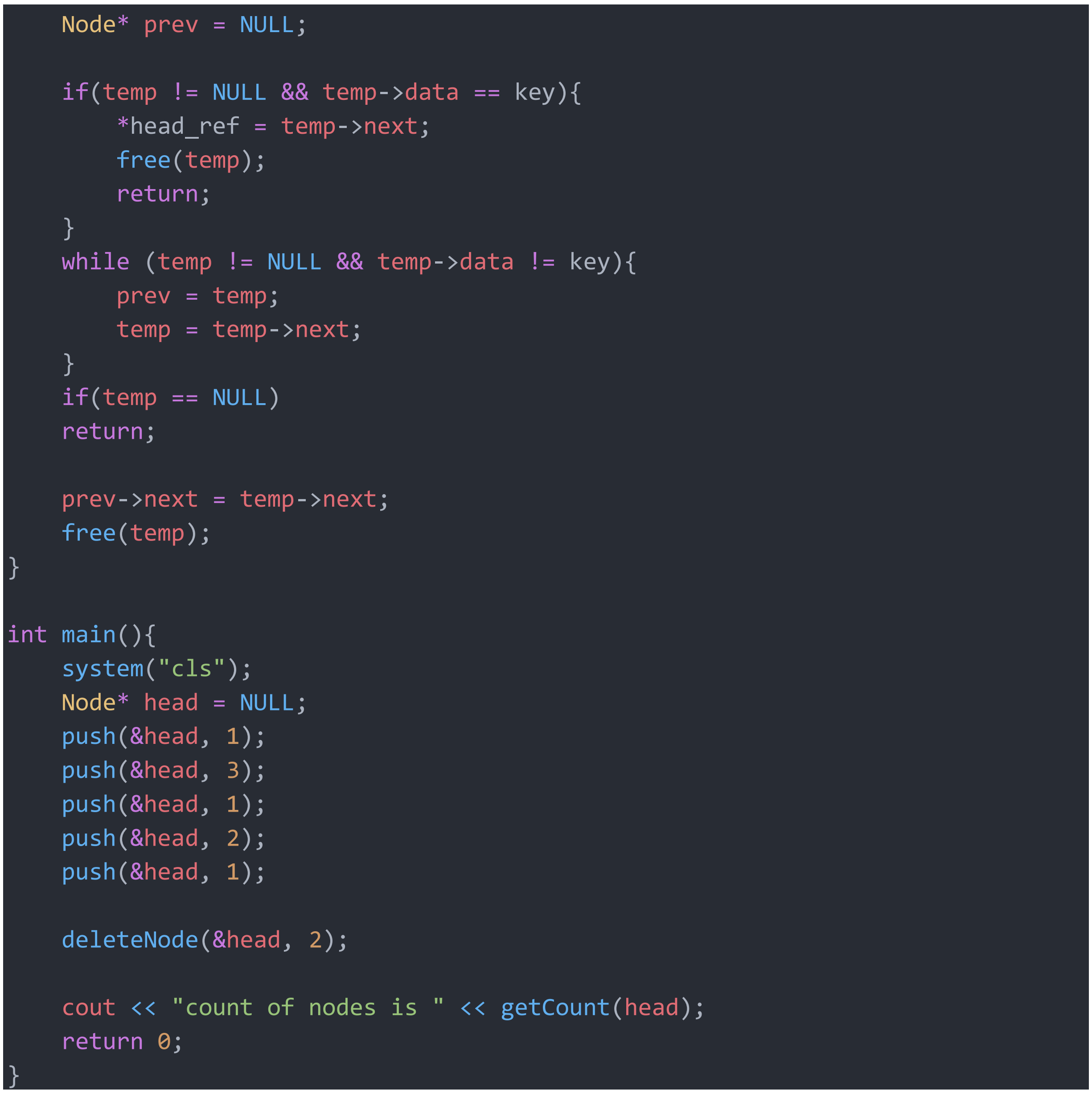
- **Output**



```
 Deleting linked list
 Linked list deleted
PS D:\SMT 2\PD 2\Prak4> 
```

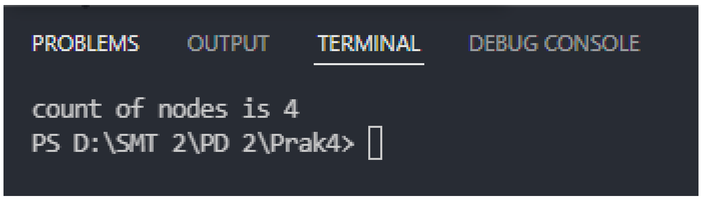
- **6a.cpp (Iterative)**

```cpp
#include <bits/stdc++.h>
using namespace std;

class Node{
    public:
    int data;
    Node* next;
};

void push(Node** head_ref, int new_data){
    Node* new_node = new Node();
    new_node->data = new_data;
    new_node->next = (*head_ref);
    (*head_ref) = new_node;
}

int getCount(Node* head){
    int count = 0;
    Node* current = head;
    while(current != NULL){
        count++;
        current = current->next;
    }
    return count;
}

void deleteNode(Node** head_ref, int key){
    Node* temp = *head_ref;
```



```
    Node* prev = NULL;

    if(temp != NULL && temp->data == key){
        *head_ref = temp->next;
        free(temp);
        return;
    }
    while (temp != NULL && temp->data != key){
        prev = temp;
        temp = temp->next;
    }
    if(temp == NULL)
    return;

    prev->next = temp->next;
    free(temp);
}

int main(){
    system("cls");
    Node* head = NULL;
    push(&head, 1);
    push(&head, 3);
    push(&head, 1);
    push(&head, 2);
    push(&head, 1);

    deleteNode(&head, 2);

    cout << "count of nodes is " << getCount(head);
    return 0;
}
```

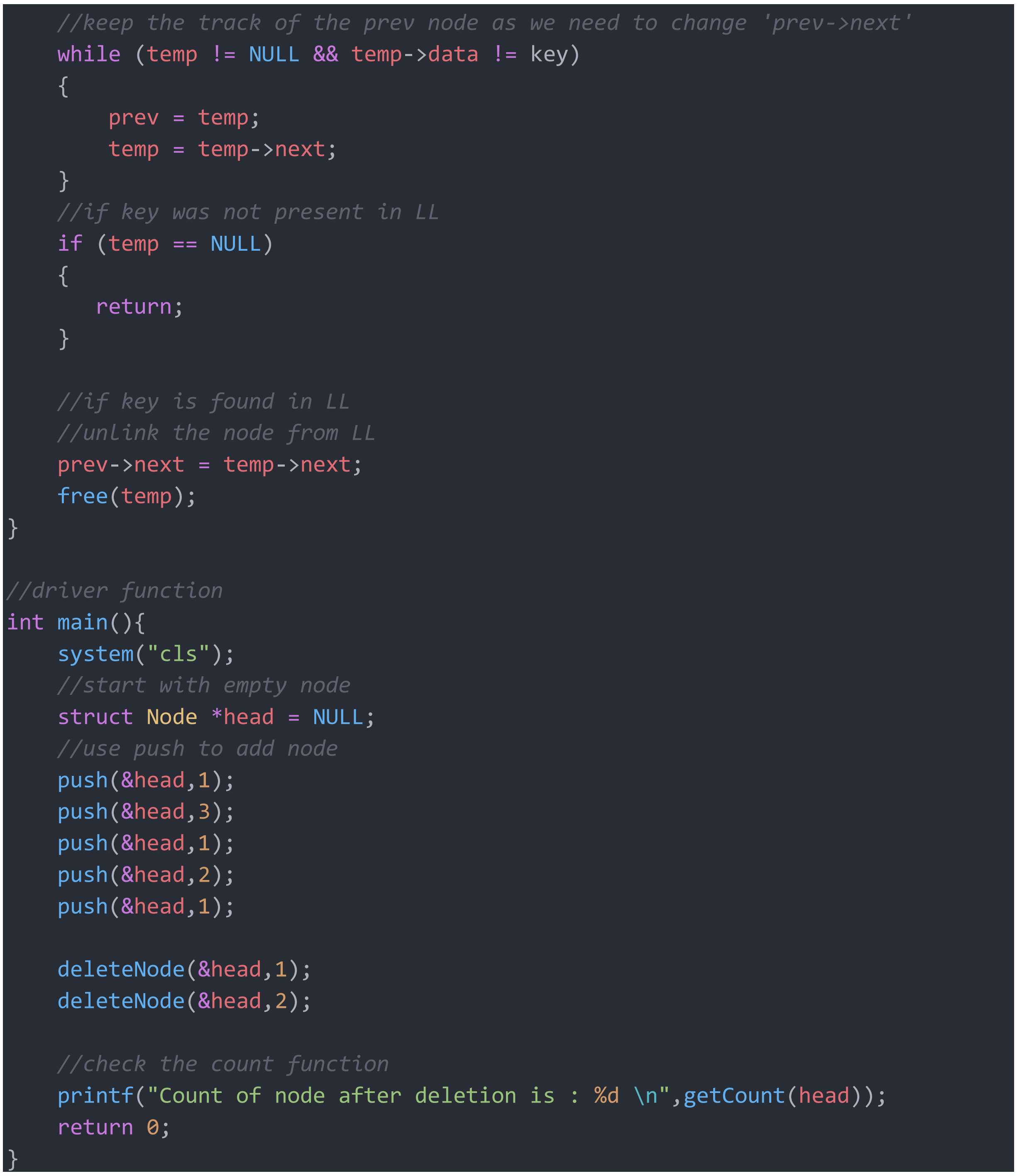
- **Output**



```
count of nodes is 4
PS D:\SMT 2\PD 2\Prak4> 
```

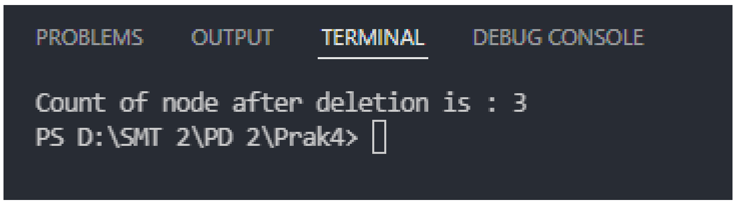
- **6b.c (Recursive)**

```c
#include <stdio.h>
#include <stdlib.h>

struct Node
{
    int data;
    struct Node *next;
};

void push(struct Node **head_ref, int new_data){

    //allocata node
    struct Node *new_node = (struct Node*)malloc(sizeof(struct Node));

    //put in data
    new_node ->data = new_data;

    //link the new node to old head
    new_node ->next = *head_ref;

    //move the head to the new node
    *head_ref = new_node;
}

//count node
int getCount(struct Node*head){
    //base case
    if (head == NULL)
    {
        return 0;
    }
    //count is 1+count of remaining node
    return 1 + getCount(head->next);
}
void deleteNode (struct Node **head_ref, int key){
    //store heap data
    struct Node *temp = *head_ref;
    struct Node *prev = NULL;

    //if head node itself holds the key to be deleted
    if (temp != NULL && temp->data == key)
    {
        *head_ref = temp->next;
        free(temp);
        return;
    }
    //search for the key to be deleted
```



```c
    //keep the track of the prev node as we need to change 'prev->next'
    while (temp != NULL && temp->data != key)
    {
        prev = temp;
        temp = temp->next;
    }
    //if key was not present in LL
    if (temp == NULL)
    {
       return;
    }

    //if key is found in LL
    //unlink the node from LL
    prev->next = temp->next;
    free(temp);
}

//driver function
int main(){
    system("cls");
    //start with empty node
    struct Node *head = NULL;
    //use push to add node
    push(&head,1);
    push(&head,3);
    push(&head,1);
    push(&head,2);
    push(&head,1);

    deleteNode(&head,1);
    deleteNode(&head,2);

    //check the count function
    printf("Count of node after deletion is : %d \n",getCount(head));
    return 0;
}
```

- **Output**



```
Count of node after deletion is : 3
PS D:\SMT 2\PD 2\Prak4> 
```

➢ **Analisa**

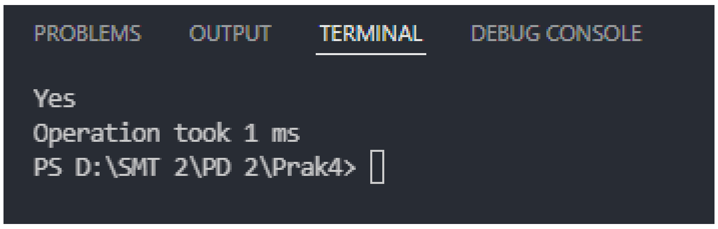
Berdasarkan percobaan tersebut diketahui bahwa pada program diatas berfungsi untuk menghitung nodes atau jarak pada sebuah SLL.

➢ **7a.cpp (Iterative)**

```
#include <bits/stdc++.h>
using namespace std;

/* Link list node */
class Node
{
    public:
    int key;
    Node* next;
};

/* Given a reference (pointer to pointer) to the head
of a list and an int, push a new node on the front
of the list. */
void push(Node** head_ref, int new_key)
{
    /* allocate node */
    Node* new_node = new Node();

    /* put in the key */
    new_node->key = new_key;

    /* link the old list off the new node */
    new_node->next = (*head_ref);

    /* move the head to point to the new node */
    (*head_ref) = new_node;
}

/* Checks whether the value x is present in linked list */
bool search(Node* head, int x)
{
    Node* current = head; // Initialize current
    while (current != NULL)
    {
        if (current->key == x)
            return true;
        current = current->next;
    }
    return false;
```

```cpp
}

/* Driver program to test count function*/
int main()
{
    system("cls");
    /* Start with the empty list */
    Node* head = NULL;
    int x = 30;

    /* Use push() to construct below list
    14->21->11->30->10 */
    push(&head, 10);
    push(&head, 30);
    push(&head, 11);
    push(&head, 21);
    push(&head, 14);

    //Start searching
    clock_t start;
    double duration;
    start = clock();

    search(head, x)? cout<<"Yes" : cout<<"No";

    duration = (clock() - start)/(double)(CLOCKS_PER_SEC/1000);
    cout<<endl<<"Operation took "<<duration<<" ms"<<endl;
    return 0;
}
```

- **Output**



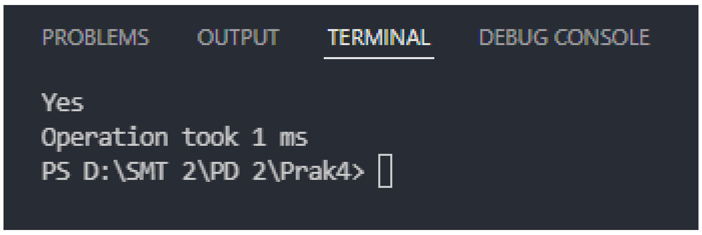
```
Yes
Operation took 1 ms
PS D:\SMT 2\PD 2\Prak4>
```

- **7b.cpp (Recursive)**

```cpp
#include <bits/stdc++.h>
using namespace std;

/* Link list node */
struct Node
```

```c
{
    int key;
    struct Node* next;
};

/* Given a reference (pointer to pointer) to the head
of a list and an int, push a new node on the front
of the list. */
void push(struct Node** head_ref, int new_key)
{
    /* allocate node */
    struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));

    /* put in the key */
    new_node->key = new_key;

    /* link the old list off the new node */
    new_node->next = (*head_ref);

    /* move the head to point to the new node */
    (*head_ref) = new_node;
}

/* Checks whether the value x is present in linked list */
bool search(struct Node* head, int x)
{
    if (head == NULL)
    return false;

    if(head->key == x)
    return true;

    return search(head->next, x);
}

/* Driver program to test count function*/
int main()
{
    system("cls");
    /* Start with the empty list */
    struct Node* head = NULL;
    int x = 21;

    /* Use push() to construct below list
    14->21->11->30->10 */
    push(&head, 10);
    push(&head, 30);
    push(&head, 11);
    push(&head, 21);
    push(&head, 14);

    //Start searching
    clock_t start;
    double duration;
    start = clock();

    search(head, x)? cout<<"Yes" : cout<<"No";

    duration = (clock() - start)/(double)(CLOCKS_PER_SEC/1000);
    cout<<endl<<"Operation took "<<duration<<" ms"<<endl;
    return 0;
}
```

➢ **Output**



```
Yes
Operation took 1 ms
PS D:\SMT 2\PD 2\Prak4>
```

➢ **Analisa**

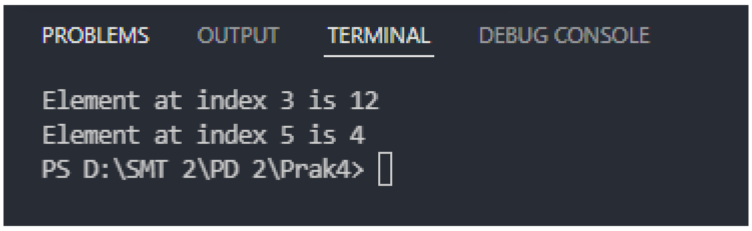
Berdasarkan percobaan tersebut dapat diketahui bahwa program tersebut berfungsi untuk melihat apakah dalam suatu linked list memiliki elemen yang dicari dan memberikan output "yes" jika benar dan "No" jika salah. Ketika dibandingkan, Iteration lebih cepat. Tetapi hasil tersebut terkadang bervariasi dan hasilnya menjadi tidak pasti.

➢ **8a.cpp**

```
#include <assert.h>
#include <bits/stdc++.h>
using namespace std;

// Link list node
class Node {
public:
    int data;
    Node* next;
};

void push(Node** head_ref, int new_data)
{
```

```cpp
    // allocate node
    Node* new_node = new Node();

    // put in the data
    new_node->data = new_data;

    // link the old list
    // off the new node
    new_node->next = (*head_ref);

    // move the head to point
    // to the new node
    (*head_ref) = new_node;
}

// Takes head pointer of
// the linked list and index
// as arguments and return
// data at index
int GetNth(Node* head, int index)
{

    Node* current = head;

    // the index of the
    // node we're currently
    // looking at
    int count = 0;
    while (current != NULL) {
        if (count == index)
            return (current->data);
        count++;
        current = current->next;
    }

    /* if we get to this line,
    the caller was asking
    for a non-existent element
    so we assert fail */
    assert(0);
}

// Driver Code
int main()
{
    system("cls");
    // Start with the
    // empty list
    Node* head = NULL;

    // Use push() to construct
    // below list
    // 1->12->1->4->1
    push(&head, 1);
    push(&head, 4);
    push(&head, 1);
    push(&head, 12);
    push(&head, 1);

    push(&head,13);
    push(&head,14);

    // Check the count
    // function
    cout << "Element at index 3 is " << GetNth(head, 3) << endl;
    cout << "Element at index 5 is " << GetNth(head, 5) << endl;
    return 0;
}
```

- **Output**

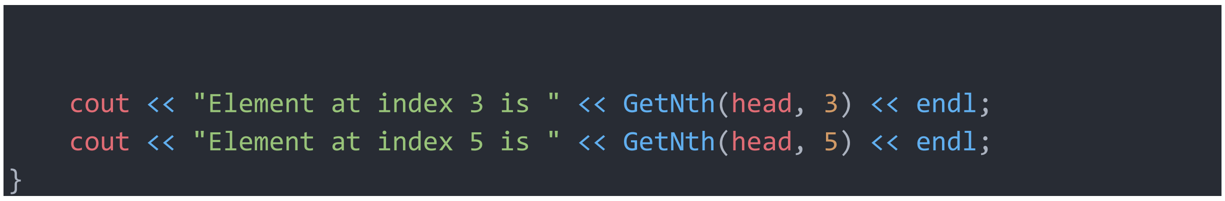


```
Element at index 3 is 12
Element at index 5 is 4
PS D:\SMT 2\PD 2\Prak4>
```

- **8b.cpp**

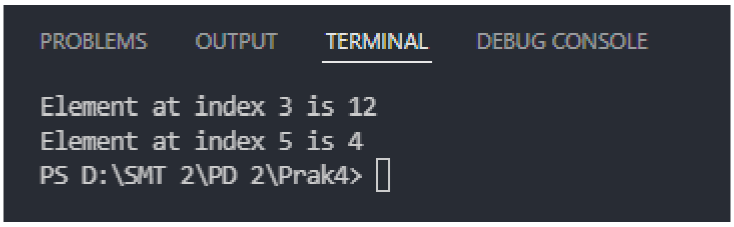
```
#include <bits/stdc++.h>
using namespace std;

/* Link list node */
struct Node {
    int data;
    struct Node* next;
};

/* Given a reference (pointer to pointer) to
    the head of a list and an int, push a
    new node on the front of the list. */
void push(struct Node** head_ref, int new_data)
{
    /* allocate node */
    struct Node* new_node
```

```c
        = (struct Node*)malloc(sizeof(struct Node));

    /* put in the data */
    new_node->data = new_data;

    /* link the old list off the new node */
    new_node->next = (*head_ref);

    /* move the head to point to the new node */
    (*head_ref) = new_node;
}

/* Takes head pointer of the linked list and index
    as arguments and return data at index. (Don't use
another variable)*/
int GetNth(struct Node* head, int n)
{
    // if length of the list is less
    // than the given index, return -1
    if (head == NULL)
        return -1;

    // if n equal to 0 return node->data
    if (n == 0)
        return head->data;

    // increase head to next pointer
    // n - 1: decrease the number of recursions until n = 0
    return GetNth(head->next, n - 1);
}

/* Driver code*/
int main()
{
    system("cls");
    /* Start with the empty list */
    struct Node* head = NULL;

    /* Use push() to construct below list
    1->12->1->4->1 */
    push(&head, 1);
    push(&head, 4);
    push(&head, 1);
    push(&head, 12);
    push(&head, 1);


    push(&head,13);
    push(&head,14);
```



```
    cout << "Element at index 3 is " << GetNth(head, 3) << endl;
    cout << "Element at index 5 is " << GetNth(head, 5) << endl;
}
```

➢ **Output**



```
Element at index 3 is 12
Element at index 5 is 4
PS D:\SMT 2\PD 2\Prak4> 
```

➢ **Analisa**

Berdasarkan percobaan diatas dapat diketahui bahwa code ini berfungsi untuk mencari nilai yang dicari dari sebuah index.Untuk Iteration menggunakan bahasa C++ dan untuk mencarinya dimulai dari nol. Untuk Recursive menggunakan bahasa C dan untuk mencarinya dimulai dari satu. Ketika index yang dicari tidak ada, Output dari Iteration akan memberi error dimana letak kesalahannya dan program berakhir. Dan untuk Output dari Recursive program akan langsung berakhir.