Review Jobsheet 4



Dosen pengampu: Randi Proska Sandra, M.Sc

Kode Kelas: 202323430158

Disusun Oleh:

Fhandy Nofalino Akhsan 23343065

PROGRAM STUDI INFORMATIKA (NK) FAKULTAS TEKNIK UNIVERSITAS NEGERI PADANG 2024

No	Baris	Petikan Source Code	Penjelasan
Program	Program		
1	5-10	<pre>struct Node { int data; struct Node *next; struct Node *prev; };</pre>	Deklarasi struktur baru dengan nama node(simpul). Next dan prev adalah variable pointer yang akan digunakan untuk mengarah kke simpul sebelum atau setelah sebuah simpul baru dibuat
1	1-2	<pre>#include <stdio.h> #include <stdlib.h></stdlib.h></stdio.h></pre>	#include <stdio.h>: Ini adalah sebuah preprocessor directive yang menyertakan file header stdio.h, yang berisi deklarasi fungsi-fungsi standar untuk input dan output seperti printf dan scanf. #include <stdlib.h>: Ini juga merupakan sebuah preprocessor directive yang menyertakan file header stdlib.h, yang berisi deklarasi fungsi-fungsi standar untuk alokasi memori, konversi, dan fungsi-fungsi umum lainnya.</stdlib.h></stdio.h>
1	10-25	<pre>void push(struct Node** head_ref, int new_data) { /* 1. allocate node */ struct Node* new_node = (struct Node*)malloc(sizeof(struct Node)); /* 2. put in the data */ new_node->data = new_data; /* 3. Make next of new node as head and previous as NULL */</pre>	void push(struct Node** head_ref, int new_data): Ini adalah deklarasi fungsi push. Fungsi ini mengambil dua parameter: head_ref, yang merupakan pointer ke pointer menuju kepala (head) dari linked list, dan new_data, yang merupakan data yang akan dimasukkan ke dalam node baru. struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));: Baris ini mengalokasikan memori

```
new_node->next =
(*head_ref);
new_node->prev = NULL;
/* 4. change prev of head
node to new node */
if ((*head_ref) != NULL)
(*head_ref)->prev =
new_node;
/* 5. move the head to
point to the new node */
(*head_ref) = new_node;
}
```

untuk node baru menggunakan fungsi malloc(). Ukuran alokasi memori sesuai dengan ukuran struktur struct Node. Fungsi malloc() mengembalikan alamat memori dari node baru yang telah dialokasikan. new node->data = new data;: Baris ini menetapkan data baru (new data) ke dalam node yang baru saja dialokasikan. new node->next = (*head ref);: Ini mengatur pointer next dari node baru untuk menunjuk ke node yang saat ini menjadi kepala dari linked list (yang ditunjuk oleh head ref). new node->prev = NULL;: Karena node baru akan menjadi kepala dari linked list, maka pointer prev dari node baru diatur menjadi NULL, menandakan bahwa tidak ada node sebelumnya. if ((*head ref) != NULL) (*head ref)->prev = new node;: Ini memeriksa apakah linked list tidak kosong. Jika tidak kosong, maka pointer prev dari node yang saat ini menjadi kepala linked list (yang ditunjuk oleh head ref) diubah untuk menunjuk ke node baru. (*head_ref) = new node;: Akhirnya, pointer head ref diubah untuk menunjuk ke node baru, menjadikannya

			sebagai kepala baru dari
			linked list.
1	26-40	void printList(struct	Source code ini merupakan
		Node* node)	implementasi dari fungsi
		{	printList yang digunakan
		struct Node* last;	untuk mencetak isi dari
		<pre>printf("\nTraversal in</pre>	suatu linked list. Fungsi ini
		forward direction \n");	menerima parameter
		while (node != NULL) {	berupa pointer ke node
		printf(" %d ", node-	pertama dari linked list.
		>data);	Pertama, fungsi mencetak
		last = node;	isi linked list secara
		<pre>node = node->next;</pre>	berurutan dari node
		}	pertama hingga terakhir
		<pre>printf("\nTraversal in</pre>	dengan menggunakan
		reverse direction \n");	perulangan while yang
		while (last != NULL) {	berjalan selama node tidak
		printf(" %d ", last-	NULL. Setiap kali iterasi,
		>data);	nilai data dari node saat ini
		<pre>last = last->prev;</pre>	dicetak, kemudian pointer
		}	node digeser ke node
		}	berikutnya. Setelah selesai mencetak
			isi linked list secara
			berurutan, fungsi
			melakukan pencetakan
			ulang dari belakang ke
			depan. Ini dilakukan
			dengan memanfaatkan
			pointer last yang telah
			disimpan saat iterasi
			pertama. Pencetakan
			dimulai dari last dan
			berlanjut mundur ke node
			sebelumnya menggunakan
			pointer prev. Proses ini
			terus dilakukan hingga
			pointer last menjadi
			NULL.
			Jadi, fungsi ini mencetak
			isi linked list dua kali,
			pertama dari awal ke akhir, dan kedua dari akhir ke
			awal.
2	1-2	<pre>#include <stdio.h></stdio.h></pre>	#include <stdio.h>:</stdio.h>
		<pre>#include <stdlib.h></stdlib.h></pre>	Mendefinisikan pustaka
			standar untuk input-output

	Γ	I	
			dalam bahasa C, yang berisi fungsi-fungsi seperti printf dan scanf.
			#include <stdlib.h>:</stdlib.h>
			Mendefinisikan pustaka
			standar untuk fungsi-
			fungsi umum dalam bahasa C, termasuk alokasi dan
			dealokasi memori dinamis
			dengan malloc dan free.
2	4-9	// Structure of the node	data: Variabel bertipe int
		struct Node	yang menyimpan nilai data
		{ : nt data:	yang akan disimpan dalam node.
		<pre>int data; struct Node *next; //</pre>	next: Pointer yang
		Pointer to next node	menunjuk ke node
		struct Node *prev; //	berikutnya dalam linked
		Pointer to previous node	list. Dalam konteks ini,
		};	linked list adalah
			rangkaian node-node yang terhubung satu sama lain.
			prev: Pointer yang
			menunjuk ke node
			sebelumnya dalam linked
			list. Ini digunakan dalam
			implementasi linked list
			dua arah (doubly linked list), di mana setiap node
			memiliki koneksi ke node
			sebelumnya dan
			sesudahnya.
2	29-35	void insertAfter(struct	prev_node: Pointer ke
		Node* prev_node, int	node sebelumnya, setelah posisi mana node baru
		new_data) {	akan disisipkan.
		/*1. check if the	new data: Data yang akan
		given prev_node is NULL */	disimpan di dalam node
		if (prev_node ==	baru yang akan disisipkan.
		NULL) {	
		printf("the	
		<pre>given previous node cannot be NULL");</pre>	
		return;	
2	50-64	void printList(struct	Pertama, variabel last dari
		Node* node)	tipe struct Node*
		{	dideklarasikan untuk
			menyimpan pointer ke

```
node terakhir dari linked
                             struct Node* last;
                             printf("\nTraversal
                                                         list.
                      in forward direction \n");
                                                         Fungsi kemudian
                                                         mencetak isi linked list
                             while (node != NULL)
                                                         secara berurutan dari awal
                      {
                                                         ke akhir dengan
                             printf(" %d ", node-
                                                         perulangan while. Dalam
                      >data);
                                                         perulangan tersebut, nilai
                             last = node;
                                                         data dari setiap node
                             node = node->next;
                                                         dicetak, kemudian pointer
                       }
                                                         node digeser ke node
                              printf("\nTraversal
                                                         berikutnya.
                      in reverse direction \n");
                                                         Setelah selesai mencetak
                             while (last != NULL)
                                                         isi linked list secara
                      {
                                                         berurutan, fungsi
                             printf(" %d ", last-
                                                         melakukan pencetakan
                      >data);
                                                         ulang dari belakang ke
                             last = last->prev;
                                                         depan. Ini dilakukan
                       }
                                                         dengan menggunakan
                      }
                                                         variabel last yang telah
                                                         disimpan saat iterasi
                                                         pertama. Pencetakan
                                                         dimulai dari last dan
                                                         berlanjut mundur ke node
                                                         sebelumnya menggunakan
                                                         pointer prev.
                                                         Proses ini terus dilakukan
                                                         hingga pointer last menjadi
                                                         NULL, sehingga semua
                                                         node telah dicetak dari
                                                         akhir ke awal.
2
           65-77
                      int main()
                                                         Pertama, sebuah pointer
                                                         head ke node pertama dari
                                                         linked list dideklarasikan
                             /* Start with the
                      empty list */
                                                         dan diinisialisasi sebagai
                             struct Node* head =
                                                         NULL. Ini menandakan
                                                         bahwa linked list masih
                      NULL;
                                                         kosong saat awalnya.
                             push(&head, 6);
                                                         Kemudian, beberapa
                             push(&head, 5);
                                                         operasi dilakukan untuk
                             push(&head, 2);
                                                         mengubah linked list:
                             insertAfter(head-
                                                         Tiga panggilan fungsi push
                      >next, 5);
                                                         digunakan untuk
                             printf("Created DLL
                                                         menambahkan node-node
                      is: ");
                                                         baru ke depan linked list.
                             printList(head);
                                                         Angka 6, 5, dan 2
                             getchar();
                                                         ditambahkan ke linked list
                             return 0;
```

	1	١ ،	
		}	secara berurutan, sehingga
			linked list akan memiliki
			urutan 2, 5, 6.
			Panggilan fungsi
			insertAfter digunakan
			untuk menyisipkan nilai 5
			setelah node kedua (node
			dengan nilai 5).
			Setelah itu, pesan "Created
			DLL is: " dicetak sebagai
			penanda, yang diikuti oleh
			pemanggilan fungsi
			printList untuk mencetak
			isi dari linked list yang
			telah dimodifikasi.
			getchar() digunakan untuk
			menunggu hingga
			pengguna menekan tombol
			Enter sebelum program
			berakhir.
			Nilai 0 dikembalikan untuk
			menandakan bahwa
			program telah berjalan
			dengan sukses dan
			berakhir tanpa ada
			masalah.
3	1-2	<pre>#include <stdio.h></stdio.h></pre>	#include <stdio.h>:</stdio.h>
		<pre>#include <stdlib.h></stdlib.h></pre>	Mendefinisikan pustaka
			standar untuk input-output
			dalam bahasa C, yang
			berisi fungsi-fungsi seperti
			printf dan scanf.
			#include <stdlib.h>:</stdlib.h>
			Mendefinisikan pustaka
			standar untuk fungsi-
			fungsi umum dalam bahasa
			C, termasuk alokasi dan
			dealokasi memori dinamis
			dengan malloc dan free.
3	5-9	// Structure of the node	data: Variabel bertipe int
		struct Node {	yang menyimpan nilai atau
		int data;	data yang ingin disimpan
		struct Node *next; //	dalam simpul (node)
		Pointer to next node	linked list.
		struct Node *prev; //	next: Pointer yang
		Pointer to previous node	menunjuk ke simpul
) ;	(node) berikutnya dalam
L	I		-

			linked list. Dengan menggunakan pointer ini, kita bisa melakukan traversal atau perjalanan ke simpul berikutnya dalam linked list. prev: Pointer yang menunjuk ke simpul (node) sebelumnya dalam linked list. Penggunaan pointer ini umumnya terdapat pada linked list tipe doubly linked list, di mana setiap simpul memiliki koneksi ke simpul sebelumnya dan sesudahnya.
3	49-62	<pre>void printList(struct Node *node) { struct Node *last = NULL; printf("\nTraversal in forward direction \n"); while (node != NULL) { printf(" %d ", node->data); last = node; node = node->next; } printf("\nTraversal in reverse direction \n"); while (last != NULL) { printf(" %d ", last->data); last = last->prev; } }</pre>	Pertama, sebuah pointer last dari tipe struct Node* dideklarasikan dan diinisialisasi sebagai NULL. Pointer ini akan digunakan untuk menyimpan alamat dari node terakhir dalam linked list. Fungsi kemudian mencetak isi linked list secara berurutan dari awal ke akhir dengan menggunakan perulangan while. Dalam perulangan tersebut, nilai data dari setiap node dicetak menggunakan printf, kemudian pointer last diupdate untuk menunjuk ke node saat ini, dan pointer node digeser ke node berikutnya. Setelah selesai mencetak isi linked list secara berurutan, fungsi melakukan pencetakan ulang dari belakang ke depan. Ini dilakukan

			dengan menggunakan variabel last yang telah disimpan saat iterasi pertama. Pencetakan dimulai dari last dan berlanjut mundur ke node sebelumnya menggunakan pointer prev. Proses ini terus dilakukan hingga pointer last menjadi NULL, sehingga semua
3	64-79	<pre>int main() { // Start with the empty list struct Node *head = NULL; // Insert 6. So linked list becomes 6->NULL append(&head, 6); // Insert 7 at the beginning. So linked list becomes 7->6->NULL push(&head, 7); // Insert 1 at the beginning. So linked list becomes 1->7->6->NULL push(&head, 1); // Insert 4 at the end. So linked list becomes 1->7->6->4->NULL append(&head, 4); printf("Created DLL is: "); printList(head); getchar(); return 0; }</pre>	akhir ke awal. Pertama, sebuah pointer head dari tipe struct Node* dideklarasikan dan diinisialisasi sebagai NULL. Ini menandakan bahwa linked list masih kosong saat awalnya. Kemudian, beberapa operasi dilakukan untuk mengubah linked list: Panggilan fungsi append digunakan untuk menambahkan node dengan nilai 6 ke akhir linked list. Sehingga linked list akan menjadi 6->NULL. Panggilan fungsi push digunakan untuk menambahkan node dengan nilai 7 di awal linked list. Sehingga linked list akan menjadi 7->6->NULL. Panggilan fungsi push kembali digunakan untuk menambahkan node dengan nilai 1 di awal linked list. Sehingga linked list akan menjadi 1->7->6->NULL. Panggilan fungsi push kembali digunakan untuk menambahkan node dengan nilai 1 di awal linked list. Sehingga linked list akan menjadi 1->7->6->NULL. Panggilan fungsi append kembali digunakan untuk menambahkan node

			dengan nilai 4 di akhir
			linked list. Sehingga linked
			list akan menjadi 1->7->6-
			>4->NULL.
			Setelah semua operasi
			-
			penambahan selesai
			dilakukan, pesan "Created
			DLL is: " dicetak sebagai
			penanda, yang diikuti oleh
			pemanggilan fungsi
			printList untuk mencetak
			isi dari linked list yang
			telah dimodifikasi.
			getchar() digunakan untuk
			menunggu hingga
			pengguna menekan tombol
			Enter sebelum program
			berakhir.
			Nilai 0 dikembalikan untuk
			menandakan bahwa
			program telah berjalan
			dengan sukses dan
			berakhir tanpa ada
			masalah.
	11 21		D / C · · ·
3	11-24	void push(struct Node	Pertama, fungsi ini
3	11-24	<pre>**head_ref, int new_data)</pre>	melakukan alokasi memori
3	11-24	• •	
3	11-24	<pre>**head_ref, int new_data)</pre>	melakukan alokasi memori
3	11-24	<pre>**head_ref, int new_data) { // Allocate node</pre>	melakukan alokasi memori untuk node baru
3	11-24	<pre>**head_ref, int new_data) { // Allocate node struct Node *new_node</pre>	melakukan alokasi memori untuk node baru menggunakan fungsi
3	11-24	<pre>**head_ref, int new_data) { // Allocate node struct Node *new_node = (struct Node</pre>	melakukan alokasi memori untuk node baru menggunakan fungsi malloc, sehingga variabel
3	11-24	<pre>**head_ref, int new_data) { // Allocate node struct Node *new_node = (struct Node *)malloc(sizeof(struct)</pre>	melakukan alokasi memori untuk node baru menggunakan fungsi malloc, sehingga variabel new_node menunjuk ke
3	11-24	<pre>**head_ref, int new_data) { // Allocate node struct Node *new_node = (struct Node *)malloc(sizeof(struct Node));</pre>	melakukan alokasi memori untuk node baru menggunakan fungsi malloc, sehingga variabel new_node menunjuk ke alamat memori yang baru dialokasikan.
3	11-24	<pre>**head_ref, int new_data) { // Allocate node struct Node *new_node = (struct Node *)malloc(sizeof(struct Node)); // Put in the data</pre>	melakukan alokasi memori untuk node baru menggunakan fungsi malloc, sehingga variabel new_node menunjuk ke alamat memori yang baru dialokasikan. Selanjutnya, nilai
3	11-24	<pre>**head_ref, int new_data) { // Allocate node struct Node *new_node = (struct Node *)malloc(sizeof(struct Node)); // Put in the data new_node->data =</pre>	melakukan alokasi memori untuk node baru menggunakan fungsi malloc, sehingga variabel new_node menunjuk ke alamat memori yang baru dialokasikan.
3	11-24	<pre>**head_ref, int new_data) { // Allocate node struct Node *new_node = (struct Node *)malloc(sizeof(struct Node)); // Put in the data new_node->data = new_data;</pre>	melakukan alokasi memori untuk node baru menggunakan fungsi malloc, sehingga variabel new_node menunjuk ke alamat memori yang baru dialokasikan. Selanjutnya, nilai new_data dimasukkan ke dalam variabel data pada
3	11-24	<pre>**head_ref, int new_data) { // Allocate node struct Node *new_node = (struct Node *)malloc(sizeof(struct Node)); // Put in the data new_node->data = new_data; // Make next of new</pre>	melakukan alokasi memori untuk node baru menggunakan fungsi malloc, sehingga variabel new_node menunjuk ke alamat memori yang baru dialokasikan. Selanjutnya, nilai new_data dimasukkan ke
3	11-24	<pre>**head_ref, int new_data) { // Allocate node struct Node *new_node = (struct Node *)malloc(sizeof(struct Node)); // Put in the data new_node->data = new_data; // Make next of new node as head and previous</pre>	melakukan alokasi memori untuk node baru menggunakan fungsi malloc, sehingga variabel new_node menunjuk ke alamat memori yang baru dialokasikan. Selanjutnya, nilai new_data dimasukkan ke dalam variabel data pada node baru yang telah dialokasikan.
3	11-24	<pre>**head_ref, int new_data) { // Allocate node struct Node *new_node = (struct Node *)malloc(sizeof(struct Node)); // Put in the data new_node->data = new_data; // Make next of new node as head and previous as NULL</pre>	melakukan alokasi memori untuk node baru menggunakan fungsi malloc, sehingga variabel new_node menunjuk ke alamat memori yang baru dialokasikan. Selanjutnya, nilai new_data dimasukkan ke dalam variabel data pada node baru yang telah
3	11-24	<pre>**head_ref, int new_data) { // Allocate node struct Node *new_node = (struct Node *)malloc(sizeof(struct Node)); // Put in the data new_node->data = new_data; // Make next of new node as head and previous as NULL new_node->next =</pre>	melakukan alokasi memori untuk node baru menggunakan fungsi malloc, sehingga variabel new_node menunjuk ke alamat memori yang baru dialokasikan. Selanjutnya, nilai new_data dimasukkan ke dalam variabel data pada node baru yang telah dialokasikan. Kemudian, pointer next dari node baru diarahkan
3	11-24	<pre>**head_ref, int new_data) { // Allocate node struct Node *new_node = (struct Node *)malloc(sizeof(struct Node)); // Put in the data new_node->data = new_data; // Make next of new node as head and previous as NULL new_node->next = *head_ref;</pre>	melakukan alokasi memori untuk node baru menggunakan fungsi malloc, sehingga variabel new_node menunjuk ke alamat memori yang baru dialokasikan. Selanjutnya, nilai new_data dimasukkan ke dalam variabel data pada node baru yang telah dialokasikan. Kemudian, pointer next dari node baru diarahkan ke node pertama (head)
3	11-24	<pre>**head_ref, int new_data) { // Allocate node struct Node *new_node = (struct Node *)malloc(sizeof(struct Node)); // Put in the data new_node->data = new_data; // Make next of new node as head and previous as NULL new_node->next = *head_ref; new_node->prev = NULL;</pre>	melakukan alokasi memori untuk node baru menggunakan fungsi malloc, sehingga variabel new_node menunjuk ke alamat memori yang baru dialokasikan. Selanjutnya, nilai new_data dimasukkan ke dalam variabel data pada node baru yang telah dialokasikan. Kemudian, pointer next dari node baru diarahkan ke node pertama (head) dari linked list yang sudah
3	11-24	<pre>**head_ref, int new_data) { // Allocate node struct Node *new_node = (struct Node *)malloc(sizeof(struct Node)); // Put in the data new_node->data = new_data; // Make next of new node as head and previous as NULL new_node->next = *head_ref; new_node->prev = NULL; // Change prev of head</pre>	melakukan alokasi memori untuk node baru menggunakan fungsi malloc, sehingga variabel new_node menunjuk ke alamat memori yang baru dialokasikan. Selanjutnya, nilai new_data dimasukkan ke dalam variabel data pada node baru yang telah dialokasikan. Kemudian, pointer next dari node baru diarahkan ke node pertama (head) dari linked list yang sudah ada sebelumnya. Pointer
3	11-24	<pre>**head_ref, int new_data) { // Allocate node struct Node *new_node = (struct Node *)malloc(sizeof(struct Node)); // Put in the data new_node->data = new_data; // Make next of new node as head and previous as NULL new_node->next = *head_ref; new_node->prev = NULL; // Change prev of head node to new node</pre>	melakukan alokasi memori untuk node baru menggunakan fungsi malloc, sehingga variabel new_node menunjuk ke alamat memori yang baru dialokasikan. Selanjutnya, nilai new_data dimasukkan ke dalam variabel data pada node baru yang telah dialokasikan. Kemudian, pointer next dari node baru diarahkan ke node pertama (head) dari linked list yang sudah ada sebelumnya. Pointer prev diatur sebagai NULL
3	11-24	<pre>**head_ref, int new_data) { // Allocate node struct Node *new_node = (struct Node *)malloc(sizeof(struct Node)); // Put in the data new_node->data = new_data; // Make next of new node as head and previous as NULL new_node->next = *head_ref; new_node->prev = NULL; // Change prev of head node to new node if (*head_ref != NULL)</pre>	melakukan alokasi memori untuk node baru menggunakan fungsi malloc, sehingga variabel new_node menunjuk ke alamat memori yang baru dialokasikan. Selanjutnya, nilai new_data dimasukkan ke dalam variabel data pada node baru yang telah dialokasikan. Kemudian, pointer next dari node baru diarahkan ke node pertama (head) dari linked list yang sudah ada sebelumnya. Pointer prev diatur sebagai NULL karena node baru akan
3	11-24	<pre>**head_ref, int new_data) { // Allocate node struct Node *new_node = (struct Node *)malloc(sizeof(struct Node)); // Put in the data new_node->data = new_data; // Make next of new node as head and previous as NULL new_node->next = *head_ref; new_node->prev = NULL; // Change prev of head node to new node if (*head_ref!= NULL)</pre>	melakukan alokasi memori untuk node baru menggunakan fungsi malloc, sehingga variabel new_node menunjuk ke alamat memori yang baru dialokasikan. Selanjutnya, nilai new_data dimasukkan ke dalam variabel data pada node baru yang telah dialokasikan. Kemudian, pointer next dari node baru diarahkan ke node pertama (head) dari linked list yang sudah ada sebelumnya. Pointer prev diatur sebagai NULL karena node baru akan menjadi node pertama,
3	11-24	<pre>**head_ref, int new_data) { // Allocate node struct Node *new_node = (struct Node *)malloc(sizeof(struct Node)); // Put in the data new_node->data = new_data; // Make next of new node as head and previous as NULL new_node->next = *head_ref; new_node->prev = NULL; // Change prev of head node to new node if (*head_ref != NULL)</pre>	melakukan alokasi memori untuk node baru menggunakan fungsi malloc, sehingga variabel new_node menunjuk ke alamat memori yang baru dialokasikan. Selanjutnya, nilai new_data dimasukkan ke dalam variabel data pada node baru yang telah dialokasikan. Kemudian, pointer next dari node baru diarahkan ke node pertama (head) dari linked list yang sudah ada sebelumnya. Pointer prev diatur sebagai NULL karena node baru akan

		// Move the head to	Setelah itu, langkah-
		point to the new node	langkah untuk mengubah
		ļ ·	-
		*head_ref = new_node;	status node pertama dan
		}	pointer head_ref
			dilakukan:
			Jika linked list tidak
			kosong (pointer head_ref
			tidak NULL), maka
			pointer prev dari node
			pertama diubah menjadi
			menunjuk ke node baru,
			karena node baru akan
			menjadi node pertama.
			Pointer head ref diarahkan
			untuk menunjuk ke node
			baru yang telah
			ditambahkan, sehingga
			node baru menjadi node
			pertama dalam linked list.
4	1-2	#include <stdio.h></stdio.h>	#include <stdio.h>:</stdio.h>
4	1-2		
		#include <stdlib.h></stdlib.h>	Mendefinisikan pustaka
			standar untuk input-output
			dalam bahasa C, yang
			berisi fungsi-fungsi seperti
			printf dan scanf.
			#include <stdlib.h>:</stdlib.h>
			Mendefinisikan pustaka
			standar untuk fungsi-
			fungsi umum dalam bahasa
			C, termasuk alokasi dan
			dealokasi memori dinamis
			dengan malloc dan free.
4	4-9	// Structure of the node	data: Variabel bertipe int
		struct Node	yang menyimpan nilai atau
		{	data yang ingin disimpan
		int data;	dalam simpul (node)
		struct Node *next; //	linked list. □ next: Pointer
		Pointer to next node	yang menunjuk ke simpul
		struct Node *prev; //	(node) berikutnya dalam
		Pointer to previous node	linked list. Dengan
		};	menggunakan pointer ini,
		J 3	kita bisa melakukan
			traversal atau perjalanan ke
			simpul berikutnya dalam
			linked list. prev: Pointer
			yang menunjuk ke simpul
			(node) sebelumnya dalam

linked list. Penggunaan pointer ini umumnya terdapat pada linked list tipe doubly linked list, di mana setiap simpul memiliki koneksi ke simpul sebelumnya dan sesudahnya. 4 66-78 int main() { Fungsi main: Fungsi ini merupakan titi /* Start with the empty list */ struct Node* head = NULL; push(&head, 6); linked list. Penggunaan pointer ini umumnya terdapat pada linked list tipe doubly linked list. Fungsi simpul memiliki koneksi ke simpul sebelumnya dan sesudahnya. Fungsi main: Fungsi ini merupakan titi masuk (entry point) dari program. Pertama, pointer head diinisialisasi dengan nilai
terdapat pada linked list tipe doubly linked list, di mana setiap simpul memiliki koneksi ke simpul sebelumnya dan sesudahnya. 4 66-78 int main() { Fungsi main: Fungsi ini merupakan titi masuk (entry point) dari program. struct Node* head = NULL; Pertama, pointer head
tipe doubly linked list, di mana setiap simpul memiliki koneksi ke simpul sebelumnya dan sesudahnya. 4 66-78 int main() { Fungsi main: Fungsi ini merupakan titi /* Start with the empty list */ struct Node* head = NULL; Pertama, pointer head
mana setiap simpul memiliki koneksi ke simpul sebelumnya dan sesudahnya. 4 66-78 int main() { Fungsi main: Fungsi ini merupakan titi /* Start with the empty list */ struct Node* head = NULL; Pertama, pointer head
memiliki koneksi ke simpul sebelumnya dan sesudahnya. 4 66-78 int main() { Fungsi main: Fungsi ini merupakan titi masuk (entry point) dari program. struct Node* head = NULL; Pertama, pointer head
simpul sebelumnya dan sesudahnya. 4 66-78 int main() { Fungsi main: Fungsi ini merupakan titi /* Start with the empty list */ struct Node* head = NULL; Pertama, pointer head
sesudahnya. 4 66-78 int main() { Fungsi main: Fungsi ini merupakan titi /* Start with the empty list */ struct Node* head = NULL; Pertama, pointer head
fungsi main: { Fungsi ini merupakan titi /* Start with the empty list */ struct Node* head = NULL; Fungsi main: Fungsi ini merupakan titi masuk (entry point) dari program. Pertama, pointer head
{ /* Start with the empty list */ struct Node* head = NULL; Fungsi ini merupakan titi masuk (entry point) dari program. Pertama, pointer head
/* Start with the empty masuk (entry point) dari program. struct Node* head = NULL; Pertama, pointer head
list */ struct Node* head = NULL; program. Pertama, pointer head
struct Node* head = NULL; Pertama, pointer head
·
push(&head, 6); diinisialisasi dengan nilai
push(&head, 5); NULL, menandakan
push(&head, 2); bahwa linked list kosong.
insertAfter(head->next, Kemudian, tiga node bard
dengan nilai data 6, 5, da
printf("Created DLL is: 2 dimasukkan ke awal
linked list menggunakan
printList(head); fungsi push.
getchar(); Selanjutnya, node baru
return 0; dengan nilai data 5
dimasukkan setelah node
kedua (dengan nilai data
menggunakan fungsi
insertAfter.
Akhirnya, isi dari linked
list dicetak menggunakan
fungsi printList.
4 11 void push(struct Node** Fungsi push:Fungsi ini
head_ref, int new_data) beroperasi pada doubly
linked list. Parameternya
adalah:head_ref: Pointer ke
pointer yang menunjuk ke
head (awal) dari doubly
linked list.new_data: Sebua
bilangan bulat yang mewak
nilai data dari node baru
yang akan dimasukkan.
4 51-65 void printList(struct Node* Fungsi printList:Fungsi ini
node) beroperasi pada doubly
{ linked list.Parameternya
struct Node* last; adalah:node: Pointer ke
printf("\nTraversal in struct Node, yang
forward direction \n"); merupakan head (awal) da
while (node != NULL) { doubly linked list.
<pre>printf(" %d ", node->data);</pre>

```
last = node;
                       node = node->next;
                       printf("\nTraversal in
                      reverse direction \n");
                       while (last != NULL) {
                       printf(" %d ", last->data);
                       last = last->prev;
                       }
1
           41-53
                      int main()
                                                         Pertama, sebuah pointer
                                                         head dari tipe struct Node*
                                                         dideklarasikan dan
                       /* Start with the empty
                                                         diinisialisasi sebagai
                      list */
                                                         NULL. Ini menandakan
                        struct Node* head = NULL;
                                                         bahwa linked list masih
                       push(&head, 6);
                                                         kosong saat awalnya.
                       push(&head, 5);
                                                         Kemudian, beberapa
                       push(&head, 2);
                                                         operasi dilakukan untuk
                       printf("Created DLL is:
                                                         mengubah linked list:
                                                         Panggilan fungsi push
                       printList(head);
                                                         digunakan untuk
                       getchar();
                                                         menambahkan node
                                                         dengan nilai 6 di awal
                       return 0;
                                                         linked list. Sehingga linked
                      }
                                                         list akan menjadi 6-
                                                         >NULL.
                                                         Panggilan fungsi push
                                                         kembali digunakan untuk
                                                         menambahkan node
                                                         dengan nilai 5 di awal
                                                         linked list. Sehingga linked
                                                         list akan menjadi 5->6-
                                                         >NULL.
                                                         Panggilan fungsi push
                                                         kembali digunakan untuk
                                                         menambahkan node
                                                         dengan nilai 2 di awal
                                                         linked list. Sehingga linked
                                                         list akan menjadi 2->5->6-
                                                         >NULL.
                                                         Setelah semua operasi
                                                         penambahan selesai
                                                         dilakukan, pesan "Created
                                                         DLL is: " dicetak sebagai
                                                         penanda, yang diikuti oleh
                                                         pemanggilan fungsi
```

	printList untuk mencetak
	isi dari linked list yang
	telah dimodifikasi.
	getchar() digunakan untuk
	menunggu hingga
	pengguna menekan tombol
	Enter sebelum program
	berakhir.
	Nilai 0 dikembalikan untuk
	menandakan bahwa
	program telah berjalan
	dengan sukses dan
	berakhir tanpa ada
	masalah.