Nomor program	Baris program	Petikan source code	penjelasan
1	5-10	Struct node (Int data; Struct node *prev; Struct node *next;);	Deklarasi struktur baru dengan nama node (simpul).next dan prev adalah variable pointer yang akan digunakan untuk mengarahkan ke simpul sebelum atau setelah sebuah simpul baru dibuat
1	10-20	Void push(struct node **head,int new_data) Struct node*newnode=(struct node*)malloc(sizeof(struct node); Newnode->data=new_data; Newnode->next=(*headref); Newnode-<prev=NULL; If(*head_ref !=NULL) (*head_ref)=newnode; (*head_ref)=newnode)}	1. Allokasi memori untuk node baru. 2. Inisialisasi data node baru dengan nilai **new_data**. 3. Mengatur pointer **next** dari node baru ke head yang sudah ada. 4. Mengatur pointer **prev** dari node baru menjadi NULL. 5. Jika linked list sudah memiliki elemen, maka head diarahkan ke node baru. 6. Akhirnya, node baru dijadikan head baru dari linked list.

1	20-30	void printList(struct Node* node)
		{
		struct Node* last;
		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;
		}
		}
			Fungsi printList ini digunakan untuk mencetak isi dari linked list, baik dari depan ke belakang maupun dari belakang ke depan. Pertama, kita menyiapkan pointer last untuk menunjuk ke node terakhir. Kemudian, kita mulai dengan mencetak isi dari depan ke belakang menggunakan perulangan while, menyimpan setiap node sebagai last. Setelah selesai, kita mencetak isi dari belakang ke depan dengan menggunakan last dan perulangan while kembali, namun kali ini dari last ke prev-nya.

1	30-35	int main()
		{
		/* Start with the empty list */
		struct Node* head = NULL;
		push(&head, 6);
		push(&head, 5);
		push(&head, 2);
		printf("Created DLL is: ");
		printList(head);
		getchar();
		return
			Pada main, kita mulai dengan membuat linked list kosong dengan head yang menunjuk ke NULL. Kemudian, kita tambahkan beberapa elemen ke linked list menggunakan fungsi push, yaitu 6, 5, dan 2. Setelah itu, kita mencetak isi linked list dengan memanggil fungsi printList. Akhirnya, kita menggunakan getchar() untuk menahan output agar tidak langsung hilang.

2	1-10	// Structure of the node
		struct Node
		{
		int data;
		struct Node *next;
		struct Node *prev;
			Setiap node memiliki dua bagian utama: data, yang menyimpan nilai dari node tersebut, dan dua pointer, yaitu next yang menunjuk ke node berikutnya dalam linked list, dan prev yang menunjuk ke node sebelumnya

2	10-20	Node* new_node = new Node();
		
		new_node->data = new_data;
		
		new_node->next = (*head_ref);
		new_node->prev = NULL;
		
		if ((*head_ref) != NULL)
		(*head_ref)->prev = new_node;
		
		(*head_ref) = new_node;
		}
			Membuat node baru.
			Mengisi data node baru dengan nilai new_data.
			Mengatur pointer next node baru menjadi head_ref (head dari linked list) dan pointer prev menjadi NULL.
			Jika linked list tidak kosong, maka mengatur pointer prev dari head sebelumnya menjadi node baru.
			Mengubah head_ref (pointer ke head) untuk menunjuk ke node baru, sehingga node baru menjadi head baru dari linked list.

2	20-25	void insertAfter(struct Node* prev_node, int new_data)
		{
		if (prev_node == NULL) {
		printf("the given previous node cannot be NULL");
		return;
		}
			fungsi memeriksa apakah node sebelumnya yang diberikan (prev_node) tidak kosong. Jika kosong, fungsi menampilkan pesan kesalahan bahwa node sebelumnya tidak dapat menjadi NULL, dan kemudian menghentikan proses.

2	25-30	struct Node* new_node = (struct
		new_node->data = new_data;
		new_node->next = prev_node->next;
		prev_node->next = new_node;
		new_node->prev = prev_node;
		if (new_node->next != NULL)
		new_node->next->prev = new_node;
		}
			Pertama, kita alokasikan memori untuk node baru.
			Kemudian, kita isi data dari node baru dengan nilai new_data.
			Pointer next dari node baru diarahkan ke node yang berada setelah prev_node.
			Pointer next dari prev_node diarahkan ke node baru.
			Pointer prev dari node baru diarahkan ke prev_node.
			Jika node baru tidak menjadi tail (tidak NULL), maka pointer prev dari node setelah node baru diarahkan ke node baru.

2	30-40	void printList(struct Node* node)
		{
		struct Node* last;
		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;
		}
		}
			Pertama, dengan menggunakan perulangan while, kode ini mencetak setiap nilai data dari setiap node dalam linked list, dimulai dari node (head) dan bergerak ke node selanjutnya dengan node = node->next.
			Kemudian, kode mencetak kembali isi linked list dari belakang ke depan dengan mengikuti pointer prev dari node terakhir (last) hingga mencapai NULL, mencetak nilai data dari setiap node yang diakses, dan bergerak ke node sebelumnya dengan last = last->prev

2	40-50	int main()
		{
		/* Start with the empty list */
		struct Node* head = NULL;
		push(&head, 6);
		push(&head, 5);
		push(&head, 2);
		insertAfter(head->next, 5);
		printf("Created DLL is: ");
		printList(head);
		getchar();
		return 0;
		}
			Dimulai dengan membuat linked list kosong dengan head yang menunjuk ke NULL.
			Tiga nilai (6, 5, dan 2) ditambahkan ke linked list menggunakan fungsi push.
			Kemudian, nilai 5 disisipkan setelah node kedua menggunakan fungsi insertAfter.
			Setelah itu, isi dari linked list dicetak menggunakan fungsi printList.
			getchar() digunakan untuk menahan output sehingga program tidak langsung berakhir.

3.	15-20	while (last->next != NULL)
		last = last->next;
		last->next = new_node;
		new_node->prev = last;
		return;
		}
			Pada langkah ini, kita menggunakan perulangan while untuk mencari node terakhir dalam linked list. Perulangan ini akan terus berlangsung selama last->next tidak NULL, yang berarti kita belum mencapai node terakhir dalam linked list. Setiap iterasi, kita memindahkan last ke node berikutnya menggunakan last = last->next.
			
			Setelah menemukan node terakhir, kita mengubah pointer next dari node terakhir tersebut untuk menunjuk ke node baru yang akan disisipkan.
			
			Kemudian, kita mengatur pointer prev dari node baru agar menunjuk ke node terakhir sebagai pendahulunya.

4.	10-15	insertBefore(struct Node** head_ref, struct Node* next_node, int new_data) {
		/*1. check if the given next_node is NULL */
		if (next_node == NULL) {
		printf("the given next node cannot be NULL");
		return;
			Pertama, fungsi memeriksa apakah node berikutnya yang diberikan (next_node) tidak kosong. Jika kosong, fungsi menampilkan pesan kesalahan bahwa node berikutnya tidak boleh NULL, dan kemudian menghentikan proses lebih lanjut dengan menggunakan return;.

4	20-30	/* 2. allocate new node */
		struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));
		/* 3. put in the data */
		new_node->data = new_data;
		/* 4. Make prev of new node as prev of next_node */
		new_node->prev = next_node->prev;
		/* 5. Make the prev of next_node as new_node */
		next_node->prev = new_node;
		/* 6. Make next_node as next of new_node */
		new_node->next = next_node;
		/* 7. Change next of new_node's previous node */
		if (new_node->prev != NULL)
		new_node->prev->next = new_node;
		/* 8. If the prev of new_node is NULL, it will be
		the new head node */
		else
		(*head_ref) = new_node;
		}
			Pertama, kita alokasikan memori untuk node baru.
			Kemudian, kita isi data dari node baru dengan nilai new_data.
			Pointer prev dari node baru diatur agar menunjuk ke node sebelumnya dari node berikutnya (next_node).
			Pointer prev dari node berikutnya (next_node) diubah agar menunjuk ke node baru.
			Pointer next dari node baru diarahkan ke next_node.
			Jika node sebelumnya dari node baru tidak NULL (artinya node baru bukan akan menjadi head baru), maka kita ubah pointer next dari node sebelumnya agar menunjuk ke node baru.
			Jika node sebelumnya dari node baru adalah NULL, maka node baru akan menjadi head baru dari linked list dan head_ref akan diubah untuk menunjuk ke node baru.