Menganalisis 4 buah program percobaan yang terdapat pada Jobsheet4 tentang Doubly Linked List.

Nomor Program	Baris Program	Petikan Source Code	Penjelasan

1	10-15	struct Node
		{
		int data ;
		struct Node *next ; // Pointer to the next node
		struct Node *prev ; // Pointer to the previous node
		} ;
			Kode yang diberikan tersebut men definisikan sebuah struct bernama "Node" , yang merepresentasikan sebuah node dalam sebuah linked list berganda (doubly linked list) .
			Setiap node memiliki sebuah variabel integer yang disebut "data" yang berisi nilai dari node tersebut, dan dua variabel pointer bertipe "struct Node *": "next" dan "prev", yang menunjukkan ke node selanjutnya dan sebelumnya dalam linked list secara berturut-turut.

	17	void push (struct Node** head_ref , int new_data)	Fungsi "push" yang diberikan merupakan sebuah fungsi yang menambahkan sebuah node baru ke awal dari sebuah linked list. Fungsi ini memiliki dua parameter yaitu “head_ref “ dan “new_data” fungsi ini akan membuat sebuah node baru , mengisinya dengan "new_data" , mengatur pointer "next" dari node tersebut ke node yang saat ini menjadi head dari linked list , dan mengatur head dari linked list ke node baru tersebut, sehingga node baru menjadi head dari linked list.
	20	struct Node* new_node = (struct Node*) malloc (sizeof(struct Node));	Fungsi "malloc" digunakan untuk mengalokasikan sejumlah byte dalam memori. Parameter "sizeof(struct Node)" digunakan untuk menentukan jumlah byte yang diperlukan untuk mengalokasikan ruang memori yang cukup untuk sebuah node dengan tipe data "struct Node". Setelah ruang memori dialokasikan , alamat awal dari blok memori tersebut akan dikembalikan oleh fungsi "malloc" sebagai pointer bertipe "void *". Karena kita ingin menggunakannya sebagai sebuah node, maka pointer tersebut harus dicasting ke tipe "struct Node *" agar sesuai dengan tipe data node. Dalam kode tersebut, hasil dari casting pointer tersebut disimpan dalam sebuah pointer bertipe "struct Node *" yang diberi nama

			"new_node". Setelah ini, kita dapat mengakses setiap elemen dari node baru menggunakan pointer "new_node".
	32	void printList (struct Node* node)	Fungsi "printList" yang diberikan adalah sebuah fungsi yang digunakan untuk menampilkan isi dari sebuah linked list ke layar. Fungsi ini memiliki satu parameter yaitu "node", yang merupakan pointer ke head dari linked list yang akan ditampilkan. Fungsi ini akan menampilkan isi dari setiap node pada linked list secara berurutan.
	51	struct Node* head= NULL;	Kode ini mendeklarasikan sebuah pointer ke node bertipe "struct Node" dengan nama "head", dan menginisialisasinya dengan nilai NULL. Pointer "head" biasanya digunakan untuk menunjuk ke head atau elemen pertama dari sebuah linked list. Dalam inisialisasi ini, pointer "head" diberi nilai NULL yang menunjukkan bahwa linked list saat ini kosong, karena tidak ada elemen yang ditunjuk oleh head. Ketika elemen pertama ditambahkan ke linked list, pointer "head" akan di-update untuk menunjuk ke elemen pertama tersebut.
2	22	new_node->data = new_data;	Fungsi ini digunakan untuk mengisi nilai data pada sebuah node baru dalam struktur data linked list. Fungsi "new_node" adalah variabel yang menyimpan alamat memori dari node baru yang akan dibuat, dan "new_data" adalah nilai data baru yang akan dimasukkan ke dalam node tersebut. Dengan menggunakan sintaksis "new_node->data" , kode tersebut mengakses elemen data di dalam node baru kemudian mengisinya dengan nilai "new_data" .
	25	new_node->next = (*head_ref); new_node->prev = NULL;	new_node->next = (*head_ref);: Baris ini bertujuan menetapkan node berikutnya dari new_node sebagai node yang saat ini menjadi kepala dari daftar (*head_ref).

			Dengan kata lain, new_node akan ditambahkan pada posisi sebelum *head_ref. Sedangkan new_node->prev = NULL;: Baris ini bertujuan menetapkan node sebelumnya dari new_node sebagai NULL . Hal ini dilakukan karena new_node ditambahkan pada posisi pertama dalam daftar , sehingga tidak ada node sebelumnya. Dengan menetapkan prev sebagai NULL, artinya new_node ialah node pertama dalam daftar.
	35	void insertAfter(struct Node* prev_node, int new_data)	Fungsi insertAfter adalah sebuah fungsi dalam bahasa pemrograman C berfungsi untuk menambahkan sebuah node baru pada suatu daftar (linked list) setelah sebuah node tertentu yang sudah ada di dalam daftar. Fungsi ini akan menambahkan sebuah node baru dengan data new_data setelah node yang sudah ada dengan alamat prev_node.
	47	new_node->next = prev_node->next;	Baris kode new_node->next = prev_node->next; memiliki fungsi untuk menetapkan node yang berada di depannya (yang ditunjuk oleh pointer next dari prev_node) sebagai node berikutnya dari new_node. Dengan kata lain, kode tersebut akan menempatkan new_node pada posisi setelah prev_node dalam suatu daftar linked list.

	56-64	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;
		}
			Fungsi printList adalah sebuah fungsi dalam bahasa pemrograman C yang berfungsi untuk menampilkan isi dari suatu daftar linked list. Fungsi ini menerima sebuah argumen berupa pointer ke node pertama dalam daftar linked list, yang akan digunakan sebagai titik awal untuk menelusuri seluruh node dalam daftar.
			Proses yang dilakukan oleh fungsi ini adalah sebagai berikut:
			1. Mencetak pesan "Traversal in forward direction" untuk menandakan bahwa proses ini akan menampilkan isi daftar linked

			list secara berurutan. 2. Menelusuri daftar linked list dengan menggunakan sebuah loop while, dimulai dari node pertama yang diberikan sebagai argumen. Pada setiap iterasi loop, fungsi ini akan mencetak data yang terdapat pada node saat ini, menggunakan perintah printf(" %d ", node->data);. Setelah mencetak data dari node saat ini, fungsi ini akan menetapkan pointer last ke node saat ini dan menggeser pointer node ke node berikutnya dalam daftar, sehingga loop dapat melanjutkan iterasinya ke node selanjutnya. Setelah loop selesai, daftar linked list telah ditelusuri dari awal hingga akhir, dan data dari setiap node telah dicetak ke layar. Perlu dicatat bahwa fungsi ini hanya menampilkan isi daftar linked list secara berurutan dan tidak mengubah isi dari daftar linked list itu sendiri.
3	30	void append(struct Node** head_ref, int new_data)	Fungsi append adalah sebuah fungsi dalam bahasa pemrograman C yang berfungsi untuk menambahkan sebuah node baru pada akhir suatu daftar linked list. Fungsi ini menerima dua argumen, yaitu pointer ke pointer node pertama dalam daftar linked list (head_ref), dan data yang akan dimasukkan ke dalam node baru (new_data).
	33-34	struct Node* new_node = (struct Node*)malloc(sizeof(struct Node)); struct Node* last = *head_ref; /* used in step 5*/	Dalam kode struct Node* new_node = (structNode*) malloc (sizeof(struct Node));, malloc() digunakan untuk mengalokasikan memori dinamis untuk membuat sebuah node baru dalam linked list. Ukuran memori yang dialokasikan ditentukan oleh sizeof(struct Node), yang menandakan bahwa kita ingin mengalokasikan memori sebesar ukuran dari struktur Node. Setelah memori telah dialokasikan, malloc() akan mengembalikan pointer ke lokasi memori yang baru dialokasikan. Pointer ini akan

			disimpan ke variabel new_node, yang kemudian akan digunakan untuk mengakses node baru dalam linked list. Dalam kode struct Node* last = *head_ref; /* used in step 5*/, variabel last digunakan untuk menyimpan alamat dari node terakhir dalam linked list. Pointer head_ref yang merupakan pointer ke node pertama dalam linked list, digunakan untuk menelusuri linked list hingga mencapai node terakhir. Variabel last kemudian menyimpan alamat dari node terakhir tersebut, sehingga kita dapat menambahkan node baru ke akhir linked list.

	42-45	if (*head_ref == NULL) {
		new_node->prev = NULL;
		*head_ref = new_node;
		return;
			ika kondisi *head_ref == NULL terpenuhi (artinya linked list masih kosong), maka langkah-langkah yang dilakukan adalah:
			1. Mengatur pointer prev dari node baru (new_node) ke NULL, karena node baru akan menjadi node pertama dalam linked list.
			Menetapkan pointer head_ref ke node baru (new_node), karena node baru akan menjadi node pertama dalam linked list.
			2. Menghentikan eksekusi fungsi dengan menggunakan perintah return.
			Secara keseluruhan, jika linked list awalnya kosong, maka node baru yang dibuat dengan fungsi malloc akan menjadi node pertama dalam linked list dan fungsi akan berhenti setelah node baru ditambahkan ke dalam linked list.

	66-68	while (last != NULL) {
		printf(" %d ", last->data);
		last = last->prev;
			Pada baris pertama , variabel last yang merepresentasikan node terakhir dalam linked list, dicek apakah bernilai NULL atau tidak. Jika variabel last masih mengacu pada node dalam linked list, maka langkah-langkah berikut dilakukan:
			Mencetak nilai data dari node yang diwakili oleh last menggunakan perintah printf.
			Mengubah nilai last menjadi node sebelumnya dalam linked list dengan cara last = last->prev.
			Langkah-langkah tersebut akan

			terus diulang hingga seluruh node dalam linked list telah dicetak. Secara keseluruhan, kode tersebut digunakan untuk mencetak isi linked list secara terbalik, dengan menelusuri linked list dari node terakhir hingga node pertama, dan mencetak nilai data dari setiap node.

	71-90	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;
		}
			Kode tersebut membuat sebuah linked list kosong dengan menetapkan pointer head ke NULL. Kemudian, Beberapa elemen baru ditambahkan ke dalam linked list menggunakan fungsi push dan append, dan seluruh elemen dalam linked list dicetak menggunakan fungsi printList.
			Lebih spesifiknya, kode tersebut melakukan hal-hal berikut:
			1. Membuat sebuah linked list kosong dengan menetapkan pointer head ke NULL.
			2. Menambahkan elemen baru 6 ke dalam linked list menggunakan fungsi append.
			3. Menambahkan elemen baru 7 ke awal linked list menggunakan fungsi push.
			4. Menambahkan elemen baru 1 ke awal linked list menggunakan fungsi push.
			5. Menambahkan elemen baru 4 ke akhir linked list menggunakan fungsi append.
			6. Mencetak seluruh elemen dalam linked list menggunakan fungsi printList.
			7. Menunggu input dari pengguna dengan menggunakan perintah getchar.
			8.Mengakhiri program dengan mengembalikan nilai 0.
			Secara keseluruhan , kode tersebut menunjukkan bagaimana sebuah linked list dapat dibuat dan diubah dengan menambahkan elemen baru ke dalamnya , dan bagaimana elemen dalam linked list dapat dicetak menggunakan fungsi printList.

4	14-23	void push(struct Node** head_ref, int new_data)	Fungsi push di atas merupakan implementasi untuk menambahkan

		{
		struct Node* new_node = (struct Node*)malloc(sizeof(struct 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;
		}
			sebuah elemen baru ke awal dari sebuah doubly linked list. Fungsi ini menerima dua parameter yaitu head_ref dan new_data, yang merepresentasikan pointer ke head dari linked list dan nilai data dari elemen baru yang akan ditambahkan.
			Fungsi push bertujuan untuk menambahkan sebuah elemen baru ke awal dari sebuah doubly linked list dengan memperbarui pointer prev dari elemen pertama sebelumnya dan menetapkan pointer next dari elemen baru ke elemen pertama sebelumnya.

	25	void insertBefore(struct Node** head_ref, struct Node* next_node, int new_data)	Fungsi insertBefore adalah fungsi untuk menambahkan sebuah node baru sebelum suatu node tertentu pada doubly linked list . Fungsi ini menerima tiga parameter, yaitu head_ref yang merupakan pointer ke head dari linked list, next_node yang merupakan pointer ke node setelah node baru yang akan ditambahkan, dan new_data yang merupakan nilai data dari node baru yang akan ditambahkan.
	43-44	if (new_node->prev != NULL) new_node->prev->next = new_node;	Baris kode if (new_node->prev != NULL) new_node->prev->next = new_node; berfungsi untuk mengatur pointer next dari node sebelumnya pada doubly linked list. Jika node baru yang telah ditambahkan memiliki node sebelumnya, maka pointer next dari node sebelumnya harus diatur sehingga mengarah ke node baru tersebut. Jika node baru tersebut adalah elemen pertama dari linked list, maka pointer prev dari node baru harus diatur ke NULL sehingga menjadi head dari linked list.

	50-64	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;
		}
			Fungsi tersebut adalah untuk mencetak isi dari doubly linked list dari depan ke belakang dan dari belakang ke depan.
			Pertama, fungsi membuat pointer 'last' yang bertujuan untuk menyimpan alamat dari node terakhir pada doubly linked list. Kemudian fungsi mencetak semua isi dari linked list dari awal ke

		printf("\nTraversal in reverse direction \n");
		while (last != NULL) {
		printf(" %d ", last->data);
		last = last->prev;
		}
		}
			akhir, dimulai dari head hingga akhir list. Setelah itu, 'last' digunakan kembali untuk mencetak semua isi dari linked list dari akhir ke awal, dimulai dari node terakhir hingga head.

	67-79	int main()
		{
		/* Start with the empty list */
		struct Node* head = NULL;
		push(&head, 7);
		push(&head, 1);
		push(&head, 4);
		insertBefore(&head, head->next, 8);
		printf("Created DLL is: ");
		printList(head);
		getchar();
		return 0;
		}
			Fungsi main ini menginisialisasi sebuah doubly linked list kosong dengan membuat pointer 'head' yang bernilai NULL. Kemudian, beberapa operasi dilakukan pada linked list tersebut, yaitu:
			1. Push node dengan nilai data 7 ke depan linked list menggunakan fungsi 'push'.
			2. Push node dengan nilai data 1 ke depan linked list menggunakan fungsi 'push'.
			3. Push node dengan nilai data 4 ke depan linked list menggunakan fungsi 'push'.
			4. Memasukkan node baru dengan nilai data 8 sebelum node kedua pada linked list menggunakan fungsi 'insertBefore'.
			5. Mencetak isi dari linked list menggunakan fungsi 'printList'.
			6. Mengambil input karakter agar program tidak langsung keluar.
			7. Mengembalikan nilai 0, menandakan bahwa program berjalan dengan sukses.
			Hasil dari program ini adalah sebuah doubly linked list dengan isi sebagai berikut: 4 8 1 7, yang dicetak dua kali, yaitu dari depan ke belakang dan dari belakang ke depan.