1. Insertion at Front

Nomor Program	Baris Program	Petikan source code	Penjelasan
1	1 - 2	`#include <stdio.h>` `#include <stdlib.h>`	Kode #include <stdio.h> menyertakan file header yang memberikan akses kepada fungsi input-output standar C seperti printf() dan scanf(), sedangkan #include <stdlib.h> memberikan akses kepada fungsi-fungsi untuk alokasi dan dealokasi memori dinamis seperti malloc() dan free()
2	4 - 9	`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

3	11 - 31	`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 NULL;`
		`  new_node->prev = (*head_ref);`
		`  new_node->next = NULL;`
		
		`  // 4.change prev of head node to new node`
		`  if ((*head_ref) != NULL)`
		`  {`
		`  (*head_ref)->next = new_node;`
		`  }`
		
		`  // 5.Move the head to point the new node`
		`  (*head_ref) = new_node;`
		`}`
			Fungsi push menambahkan sebuah elemen baru ke depan Doubly Linked List (DLL). Langkah-langkahnya meliputi alokasi memori untuk node baru, penyisipan data baru, pengaturan pointer prev dan next, serta penyesuaian pointer next pada node sebelumnya jika DLL tidak kosong. Akhirnya, pointer head_ref diubah untuk menunjuk ke node baru yang baru ditambahkan, menjadikannya node pertama dalam DLL.

4	33 - 50	`void printlist(struct Node* node)`
		`{`
		`  struct Node* last;`
			Fungsi printlist mencetak isi Doubly Linked List (DLL) secara berurutan dari depan ke belakang, dan kemudian mencetaknya dari

		printf("\n Traversal in forward direction \n");
		while(node != NULL)
		{
		printf(" %d ",node->data);
		last = node;
		node = node->next;
		}
		
		printf("\n Traversal in reverse direction \n");
		while(last != NULL)
		{
		printf(" %d ",last->data);
		last = last->prev;
		}
		}
			belakang ke depan. Ini dilakukan dengan mengiterasi melalui DLL dua kali: pertama dari depan ke belakang dan kedua dari belakang ke depan. Selama iterasi pertama, elemen-elemen DLL dicetak dari node pertama hingga terakhir. Selama iterasi kedua, elemen-elemen dicetak dari node terakhir hingga node pertama, memberikan pandangan menyeluruh tentang isi DLL.

5.	52 - 63	int main()
		{
		//start with an empty list
		struct Node* head = NULL;
		push(&head, 6);
		push(&head, 5);
		push(&head, 2);
		printf("created DLL are:");
		printlist(head);
		getchar();
		return 0;
		}
			Dalam fungsi main, program membuat sebuah Doubly Linked List (DLL) kosong dan kemudian menambahkan tiga elemen ke dalamnya dengan nilai 6, 5, dan 2 menggunakan fungsi push. Setelah itu, isi DLL dicetak, dan program menunggu masukan pengguna sebelum berakhir.

2. Insertion After Given Node

Nomor Program	Baris Program	Petikan source code	Penjelasan
1	1 - 2	#include <stdio.h> #include <stdlib.h>	Kode #include <stdio.h> menyertakan file header yang memberikan akses kepada fungsi input-output standar C seperti printf() dan scanf(), sedangkan #include <stdlib.h> memberikan akses kepada fungsi-fungsi untuk alokasi dan dealokasi memori dinamis seperti malloc() dan free()
2	4 - 9	struct Node { int data; struct Node*	Deklarasi struktur baru dengan nama node (simpul). Next dan prev

		prev; struct Node* next; };	adalah variable pointer yang akan digunakan untuk mengarahkan ke simpul sebelum atau setelah sebuah simpul baru dibuat

3	11 - 31	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 NULL;
		new_node->prev = (*head_ref);
		new_node->next = NULL;
		
		// 4.change prev of head node to new node
		if ((*head_ref) != NULL)
		{
		(*head_ref)->next = new_node;
		}
		
		// 5.Move the head to point the new node
		(*head_ref) = new_node;
		}
			Fungsi push menambahkan sebuah elemen baru ke depan Doubly Linked List (DLL). Langkah-langkahnya meliputi alokasi memori untuk node baru, penyisipan data baru, pengaturan pointer prev dan next, serta penyesuaian pointer next pada node sebelumnya jika DLL tidak kosong. Akhirnya, pointer head_ref diubah untuk menunjuk ke node baru yang baru ditambahkan, menjadikannya node pertama dalam DLL.

4	33 - 62	void insertafter(struct Node* prev_node, int new_data)
		{
		// 1.check if the given prev node is NULL
		if(prev_node == NULL)
		{
		printf("the given previous node cannot be NULL");
		return;
		}
		
		// 2.allocate new node
		struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));
		
		// 3.Put in the data
		new_node->data = new_data;
			Fungsi insertafter menyisipkan elemen baru ke dalam Doubly Linked List (DLL) setelah node tertentu (prev_node). Langkah-langkahnya mencakup memeriksa apakah prev_node tidak NULL, mengalokasikan memori untuk node baru, menetapkan data baru, mengatur pointer next dan prev dari node baru dan node sebelumnya, serta memperbarui pointer prev dari node setelah node baru jika ada.

		`// 4.make next of new node as next of prev node`
		`new_node->next = prev_node->next;`
		
		`// 5.make next of prev node as new_node`
		`prev_node->next = new_node;`
		
		`// 6.Make prev node as previous of new_node`
		`new_node->prev = prev_node;`
		
		`// 7.change previous of new_node's next node`
		`if (new_node->next != NULL)`
		`{`
		`new_node->next->prev = new_node;`
		`}`
		`}`	

5	65 - 82	`void printlist(struct Node* node)`
		`{`
		`struct Node* last;`
		`printf("\n Traversal in forward direction \n");`
		`while(node != NULL)`
		`{`
		`printf(" %d ",node->data);`
		`last = node;`
		`node = node->next;`
		`}`
		
		`printf("\n Traversal in reverse direction \n");`
		`while(last != NULL)`
		`{`
		`printf(" %d ",last->data);`
		`last = last->prev;`
		`}`
		`}`
			Fungsi printlist mencetak isi Doubly Linked List (DLL) secara berurutan dari depan ke belakang, dan kemudian mencetaknya dari belakang ke depan. Ini dilakukan dengan mengiterasi melalui DLL dua kali: pertama dari depan ke belakang dan kedua dari belakang ke depan. Selama iterasi pertama, elemen-elemen DLL dicetak dari node pertama hingga terakhir. Selama iterasi kedua, elemen-elemen dicetak dari node terakhir hingga node pertama, memberikan pandangan menyeluruh tentang isi DLL.

6	84 - 96	`int main()`
		`{`
		`//start with an empty list`
		`struct Node* head = NULL;`
		`push(&head, 6);`
			Dalam fungsi main, program dimulai dengan membuat sebuah Doubly Linked List (DLL) kosong, kemudian menambahkan tiga elemen baru ke dalamnya dengan nilai 6, 5, dan 2 menggunakan

		push(&head, 5);
		push(&head, 2);
		insertafter(head->prev, 3);
		printf("created DLL are:");
		printlist(head);
		getchar();
		return 0;
		}
			fungsi push. Setelah itu, menggunakan fungsi insertafter, sebuah elemen baru dengan nilai 3 dimasukkan setelah node kedua dalam DLL. Selanjutnya, isi DLL dicetak menggunakan fungsi printlist, dan program menunggu masukan dari pengguna sebelum berakhir.

3. Insertion At The End

Nomor Program	Baris Program	Petikan source code	Penjelasan
1	1 - 2	#include <stdio.h> #include <stdlib.h>	Kode #include <stdio.h> menyertakan file header yang memberikan akses kepada fungsi input-output standar C seperti printf() dan scanf(), sedangkan #include <stdlib.h> memberikan akses kepada fungsi-fungsi untuk alokasi dan dealokasi memori dinamis seperti malloc() dan free()
2	4 - 9	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

3	11 - 31	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 NULL;
		new_node->prev = (*head_ref);
		new_node->next = NULL;
		
		// 4.change prev of head node to new node
		if ((*head_ref) != NULL)
		{
			Fungsi push menambahkan sebuah elemen baru ke depan Doubly Linked List (DLL). Langkah-langkahnya meliputi alokasi memori untuk node baru, penyisipan data baru, pengaturan pointer prev dan next, serta penyesuaian pointer next pada node sebelumnya jika DLL tidak kosong. Akhirnya, pointer head_ref diubah untuk menunjuk ke node baru yang baru ditambahkan, menjadikannya node pertama dalam DLL.

		(*head_ref)->next = new_node; } // 5.Move the head to point the new node (*head_ref) = new_node; }	

4	25 - 50	void append(struct Node** head_ref, int new_data)
		
		/* 1. allocate node */
		struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));
		struct Node* last = *head_ref; /* used in step 5*/
		/* 2. put in the data */
		new_node->data = new_data;
		/* 3. This new node is going to be the last node, so
		make next of it as NULL*/
		new_node->next = NULL;
		/* 4. If the Linked List is empty, then make the new
		node as head */
		if (*head_ref == NULL) {
		new_node->prev = NULL;
		*head_ref = new_node;
		return;
		}
		/* 5. Else traverse till the last node */
		while (last->next != NULL)
		last = last->next;
		/* 6. Change the next of last node */
		last->next = new_node;
		/* 7. Make last node as previous of new node */
		new_node->prev = last;
		return;
		}
			Fungsi append menambahkan elemen baru ke akhir Doubly Linked List (DLL). Langkah-langkahnya mencakup alokasi memori untuk node baru, penambahan data baru, penyesuaian pointer next dan prev node baru, serta penghubungan node baru dengan node terakhir dalam DLL. Jika DLL masih kosong, node baru dijadikan sebagai head. Dengan langkah-langkah ini, elemen baru berhasil ditambahkan ke akhir DLL.

5	51 - 65	void printlist(struct Node* node)
		{
		struct Node* last;
		printf("\n Traversal in forward direction \n");
		while(node != NULL)
		{
		printf(" %d ",node->data);
		last = node;
			Fungsi printlist mencetak isi Doubly Linked List (DLL) secara berurutan dari depan ke belakang, dan kemudian mencetaknya dari belakang ke depan. Ini dilakukan dengan mengiterasi melalui DLL dua kali: pertama dari depan ke belakang dan kedua dari belakang ke depan. Selama iterasi pertama, elemen-elemen DLL dicetak dari node pertama hingga terakhir. Selama iterasi kedua, elemen-elemen dicetak dari node terakhir

		node = node->next;
		}
		
		printf("\n Traversal in reverse direction \n");
		while(last != NULL)
		{
		printf(" %d ",last->data);
		last = last->prev;
		}
		}
			hingga node pertama, memberikan pandangan menyeluruh tentang isi DLL.

6	66 - 85	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;
		}
			Dalam fungsi main, program membuat Doubly Linked List (DLL) kosong. Beberapa operasi penambahan elemen dilakukan: 6 ditambahkan ke akhir, 7 ditambahkan ke depan, 1 juga ditambahkan ke depan, dan 4 ditambahkan ke akhir. Setelah itu, isi DLL dicetak, dan program menunggu masukan pengguna sebelum berakhir.

4. Insertion Before Given Node

Nomor Program	Baris Program	Petikan source code	Penjelasan
1	1 - 2	#include <stdio.h> #include <stdlib.h>	Kode #include <stdio.h> menyertakan file header yang memberikan akses kepada fungsi input-output standar C seperti printf() dan scanf(), sedangkan

			#include <stdlib.h> memberikan akses kepada fungsi-fungsi untuk alokasi dan dealokasi memori dinamis seperti malloc() dan free()
2	4 - 9	`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

3	11 - 31	`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 NULL;`
		`  new_node->prev = (*head_ref);`
		`  new_node->next = NULL;`
		
		`  // 4.change prev of head node to new node`
		`  if ((*head_ref) != NULL)`
		`  {`
		`  (*head_ref)->next = new_node;`
		`  }`
		
		`  // 5.Move the head to point the new node`
		`  (*head_ref) = new_node;`
		`}`
			Fungsi push menambahkan sebuah elemen baru ke depan Doubly Linked List (DLL). Langkah-langkahnya meliputi alokasi memori untuk node baru, penyisipan data baru, pengaturan pointer prev dan next, serta penyesuaian pointer next pada node sebelumnya jika DLL tidak kosong. Akhirnya, pointer head_ref diubah untuk menunjuk ke node baru yang baru ditambahkan, menjadikannya node pertama dalam DLL.

4	25 - 50	`void 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;`
		`  }`
		`  /* 2. allocate new node */`
		`  struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));`
		`  /* 3. put in the data */`
			Fungsi insertBefore bertujuan untuk menyisipkan elemen baru sebelum node tertentu dalam Doubly Linked List (DLL). Langkah-langkahnya mencakup memeriksa apakah node selanjutnya tidak NULL, alokasi memori untuk node baru, penambahan data baru, penyesuaian pointer prev dan next, serta penyesuaian pointer next dari node sebelumnya jika diperlukan. Jika node sebelumnya dari node baru NULL, node baru dijadikan sebagai head baru. Dengan langkah-langkah ini,

		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;
		}
			elemen baru berhasil disisipkan sebelum node tertentu dalam DLL.

5	51 - 65	void printlist(struct Node* node)
		{
		struct Node* last;
		printf("\n Traversal in forward direction \n");
		while(node != NULL)
		{
		printf(" %d ",node->data);
		last = node;
		node = node->next;
		}
		
		printf("\n Traversal in reverse direction \n");
		while(last != NULL)
		{
		printf(" %d ",last->data);
		last = last->prev;
		}
		}
			Fungsi printlist mencetak isi Doubly Linked List (DLL) secara berurutan dari depan ke belakang, dan kemudian mencetaknya dari belakang ke depan. Ini dilakukan dengan mengiterasi melalui DLL dua kali: pertama dari depan ke belakang dan kedua dari belakang ke depan. Selama iterasi pertama, elemen-elemen DLL dicetak dari node pertama hingga terakhir. Selama iterasi kedua, elemen-elemen dicetak dari node terakhir hingga node pertama, memberikan pandangan menyeluruh tentang isi DLL.

6	66 - 85	int main() { /* Start with the empty list */	Dalam fungsi main, program dimulai dengan membuat Doubly Linked List (DLL) kosong. Beberapa

		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;
		}
			operasi penambahan elemen dilakukan: 7, 1, dan 4 ditambahkan ke depan menggunakan fungsi push. Selanjutnya, elemen baru dengan nilai 8 disisipkan sebelum node kedua dalam DLL menggunakan fungsi insertBefore. Hasilnya, isi DLL dicetak, dan program menunggu masukan pengguna sebelum berakhir..