Program	Baris Program	Source Code	Penjelasan
1	10-14	struct Node{ int data; struct Node *next; //Pointer to next node struct Node *prev; // Pointer to previous node };	Program mendefinisikan struktur Node yang terdiri dari tiga anggota: data (integer), next (pointer ke node berikutnya), dan prev (pointer ke node sebelumnya). Struktur ini digunakan untuk membuat Linked List Ganda.
1	16-27	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 */ 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; }	Fungsi push() digunakan untuk menambahkan node baru di awal Linked List. Fungsi ini mengalokasikan memori untuk node baru, mengisi data, dan mengatur pointer next dan prev dengan benar.
1	29-42	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() digunakan untuk mencetak isi Linked List dalam dua arah: maju (forward) dan mundur (reverse). Fungsi ini melakukan traversal pada Linked List dan mencetak nilai data setiap node.

		} printf("\nTraversal in reverse direction \n"); while (last != NULL) { printf(" %d ", last->data); last = last->prev; } }	
1	44-56	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 0; }	Di dalam fungsi main(), sebuah Linked List kosong dibuat dengan head diinisialisasi dengan NULL. Kemudian, tiga node dengan nilai 6, 5, dan 2 ditambahkan ke Linked List menggunakan fungsi push().
1	50-52	printf("Created DLL is: "); printList(head); getchar();	Setelah Linked List diisi dengan tiga node, fungsi printList() dipanggil untuk mencetak isi Linked List dalam dua arah: maju (forward) dan mundur (reverse). Kemudian, program menunggu pengguna menekan tombol untuk keluar.
2	10-14	struct Node{ int data; struct Node *next; // Pointer to next node struct Node *prev; // Pointer to previous node };	Program mendefinisikan struktur Node yang terdiri dari tiga anggota: data (integer), next (pointer ke node berikutnya), dan prev (pointer ke node sebelumnya). Struktur ini digunakan untuk membuat Linked List Ganda.
2	16-33	void push(struct Node** head_ref, int new_data) { /* 1. allocate node */	Fungsi push() digunakan untuk menambahkan node baru di awal Linked List. Fungsi ini mengalokasikan memori untuk node baru, mengisi data, dan

		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 */ 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; }	mengatur pointer next dan prev dengan benar.
2	35-60	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 */	Fungsi insertAfter() digunakan untuk menambahkan node baru setelah node tertentu dalam Linked List. Fungsi ini mengalokasikan memori untuk node baru, mengisi data, dan mengatur pointer next dan prev dengan benar untuk menjaga integritas Linked List.

		struct Node* new_node = (struct Node*)malloc(sizeof(struct Node)); /* 3. put in the data */ new_node->data = new_data; /* 4. Make next of new node as next of prev_node */ new_node->next = prev_node->next; /* 5. Make the 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; }	
2	62-77	void printList (struct Node* node){ struct Node* last; printf("\nTraversal in forward direction \n"); while (node != NULL){ printf(" %d", node->data);	Fungsi printList() digunakan untuk mencetak isi Linked List dalam dua arah: maju (forward) dan mundur (reverse). Fungsi ini melakukan traversal pada Linked List dan mencetak nilai data setiap node.

		last = node; node = node->next; } printf("\nTraversal in reverse direction \n"); while (last != NULL){ printf(" %d", last->data); last = last->prev; } }	
2	79-91	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; }	Di dalam fungsi main(), sebuah Linked List kosong dibuat dengan head diinisialisasi dengan NULL. Kemudian, tiga node dengan nilai 6, 5, dan 2 ditambahkan ke Linked List menggunakan fungsi push(). Setelah itu, sebuah node baru dengan nilai 5 ditambahkan setelah node kedua menggunakan fungsi insertAfter(). Akhirnya, fungsi printList() dipanggil untuk mencetak isi Linked List dalam dua arah.
3	9-13	struct Node { int data; struct Node *next; // Pointer to next node struct Node *prev; // Pointer to previous node };	Program mendefinisikan struktur Node yang terdiri dari tiga anggota: data (integer), next (pointer ke node berikutnya), dan prev (pointer ke node sebelumnya). Struktur ini digunakan untuk membuat Linked List Ganda.
3	15-28	void push(struct Node** head_ref, int new_data){ /* 1. allocate node */ struct Node* new_node = (struct Node	Fungsi push() digunakan untuk menambahkan node baru di awal Linked List. Fungsi ini mengalokasikan memori untuk node baru, mengisi data, dan

		)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 */ 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; }	mengatur pointer next dan prev dengan benar.
3	30-52	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;	Fungsi append() digunakan untuk menambahkan node baru di akhir Linked List. Fungsi ini mengalokasikan memori untuk node baru, mengisi data, dan mengatur pointer next dan prev dengan benar untuk menjaga integritas Linked List.

		/* 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; }	
3	54-67	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);	Fungsi printList() digunakan untuk mencetak isi Linked List dalam dua arah: maju (forward) dan mundur (reverse). Fungsi ini melakukan traversal pada Linked List dan mencetak nilai data setiap node.

		last = last->prev; } }	
3	69-88	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; }	Di dalam fungsi main(), sebuah Linked List kosong dibuat dengan head diinisialisasi dengan NULL. Kemudian, node dengan nilai 6 ditambahkan di akhir menggunakan fungsi append(). Selanjutnya, node dengan nilai 7 dan 1 ditambahkan di awal menggunakan fungsi push(). Setelah itu, node dengan nilai 4 ditambahkan di akhir menggunakan fungsi append(). Akhirnya, fungsi printList() dipanggil untuk mencetak isi Linked List dalam dua arah.
4	9-13	struct Node{ int data; struct Node *next; struct Node *prev; };	Program mendefinisikan struktur Node yang terdiri dari tiga anggota: data (integer), next (pointer ke node berikutnya), dan prev (pointer ke node sebelumnya). Struktur ini digunakan untuk membuat Linked List Ganda.
4	15-23	void push(struct Node** head_ref, int new_data){ struct Node* new_node = (struct	Fungsi push() digunakan untuk menambahkan node baru di awal Linked List. Fungsi ini mengalokasikan memori untuk

		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; }	node baru, mengisi data, dan mengatur pointer next dan prev dengan benar.
4	26-48	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 */ new_node->data = new_data; /* 4. Make prev of new node as prev of next_node */ new_node->prev = next_node->prev;	Fungsi insertBefore() digunakan untuk menambahkan node baru sebelum node tertentu dalam Linked List. Fungsi ini mengalokasikan memori untuk node baru, mengisi data, dan mengatur pointer next dan prev dengan benar untuk menjaga integritas Linked List.

		/* 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; }	
4	50-63	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() digunakan untuk mencetak isi Linked List dalam dua arah: maju (forward) dan mundur (reverse). Fungsi ini melakukan traversal pada Linked List dan mencetak nilai data setiap node.

		}	
4	66-77	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; }	Di dalam fungsi main(), sebuah Linked List kosong dibuat dengan head diinisialisasi dengan NULL. Kemudian, tiga node dengan nilai 7, 1, dan 4 ditambahkan ke Linked List menggunakan fungsi push(). Setelah itu, sebuah node baru dengan nilai 8 ditambahkan sebelum node kedua menggunakan fungsi insertBefore(). Akhirnya, fungsi printList() dipanggil untuk mencetak isi