Tugas Jobsheet 4

Dosen pengampu : Randi Proska Sandra, M.Sc

Kode Kelas : 202323430158

Disusun Oleh :

Fajrul Huda Ash Shiddiq
23343063

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

No Program	Baris Program	Petikan Source Code	Penjelasan

1	5-10	`struct Node {`
		`  int data;`
		`  struct Node *next;`
		`  struct Node *prev;`
		`};`
			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 dibuat.

1	1-2	`#include <stdio.h>` `#include <stdlib.h>`	#include <stdio.h>: stdio.h berisi deklarasi fungsi-fungsi standar untuk input dan output seperti printf dan scanf. #include <stdlib.h>: stdlib.h berisi deklarasi fungsi-fungsi standar untuk alokasi memori, konversi, dan fungsi-fungsi umum lainnya.

1	10-25	`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 */`
			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 untuk node baru menggunakan fungsi malloc(). Ukuran alokasi memori sesuai dengan ukuran struktur struct Node. Fungsi malloc() mengembalikan alamat

		(*head_ref) = new_node; } 	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);: Berfungsi untuk 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;: Berfungsi untuk 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 Node* node) { struct Node* last;	Merupakan implementasi dari fungsi printList yang digunakan untuk mencetak isi dari suatu linked list. Fungsi ini menerima

		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;
		}
		}
			parameter berupa pointer ke node pertama dari linked list.
			Pertama, fungsi mencetak isi linked list secara berurutan dari node pertama hingga terakhir dengan menggunakan perulangan while yang berjalan selama node tidak NULL. Setiap kali iterasi, nilai data dari node saat ini 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	#include <stdio.h> #include <stdlib.h>	#include <stdio.h>: stdio.h berisi deklarasi fungsi-fungsi standar untuk input dan output seperti printf dan scanf. #include <stdlib.h>: stdlib.h berisi deklarasi fungsi-fungsi standar untuk alokasi memori,

			konversi, dan fungsi-fungsi umum lainnya.

2	4-9	`// Structure of the node`
		`struct Node`
		`{`
		`  int data;`
		`  struct Node *next; // Pointer to next node`
		`  struct Node *prev; // Pointer to previous node`
		`};`
			Variabel data bertipe int yang menyimpan nilai data yang akan disimpan dalam node.
			next: Pointer yang menunjuk ke node berikutnya dalam linked 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 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;`
			prev_node: Pointer ke node sebelumnya, setelah posisi mana node baru akan disisipkan.
			new_data: Data yang akan disimpan di dalam node baru yang akan disisipkan.

2	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;`
			Pertama, variabel last dari tipe struct Node* berfungsi untuk menyimpan pointer ke node terakhir dari linked list.
			Fungsi kemudian mencetak isi linked list secara berurutan dari awal ke akhir dengan perulangan while. Dalam perulangan tersebut, nilai data dari setiap node

		}
		printf("\nTraversal in reverse direction \n");
		while (last != NULL)
		{
		printf(" %d ", last->data);
		last = last->prev;
		}
		}
			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 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()
		{
		/* 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;
		}
			Pertama, sebuah pointer head ke node pertama dari linked list dideklarasikan dan diinisialisasi sebagai NULL. Ini menandakan bahwa linked list masih kosong saat awalnya.
			Kemudian, beberapa operasi dilakukan untuk mengubah linked list:
			Tiga panggilan fungsi push digunakan untuk menambahkan node-node baru ke depan linked list. Angka 6, 5, dan 2 ditambahkan ke linked list 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	`#include <stdio.h>` `#include <stdlib.h>`	#include <stdio.h>: stdio.h berisi deklarasi fungsi-fungsi standar untuk input dan output seperti printf dan scanf. #include <stdlib.h>: stdlib.h berisi deklarasi fungsi-fungsi standar untuk alokasi memori, konversi, dan fungsi-fungsi umum lainnya.

3	5-9	`// Structure of the node`
		`struct Node {`
		`  int data;`
		`  struct Node *next; // Pointer to next node`
		`  struct Node *prev; // Pointer to previous node`
		`};`
			data: Variabel bertipe int yang menyimpan nilai atau data yang ingin disimpan dalam simpul (node) linked list.
			next: Pointer yang menunjuk ke simpul (node) berikutnya dalam 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	`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;`
		`    }`
		`}`
			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 node telah dicetak dari akhir ke awal.

3	64-79	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;
		}
			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 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.

3	11-24	`void push(struct Node **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)`
		`        (*head_ref)->prev = new_node;`
		`    // Move the head to point to the new node`
		`    *head_ref = new_node;`
		`}`
			Pertama, fungsi ini 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, sehingga tidak memiliki node sebelumnya.
			Setelah itu, langkah-langkah untuk mengubah 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>` `#include <stdlib.h>`	#include <stdio.h>: stdio.h berisi deklarasi fungsi-fungsi standar untuk input dan output seperti printf dan scanf. #include <stdlib.h>: stdlib.h berisi deklarasi fungsi-fungsi standar untuk alokasi memori, konversi, dan fungsi-fungsi umum lainnya.

4	4-9	`// Structure of the node`
		`struct Node`
		`{`
		`int data;`
		`struct Node *next; //`
		`Pointer to next node`
		`struct Node *prev; //`
		`Pointer to previous node`
		`};`
			Variabel data bertipe int yang menyimpan nilai atau data yang ingin disimpan dalam simpul (node) linked list.□next: Pointer yang menunjuk ke simpul (node) berikutnya dalam 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.

4	66-78	`int main()`
		`{`
		`/* Start with the empty`
		`list */`
			Fungsi main:
			Fungsi ini merupakan titik masuk (entry point) atau fungsi yang akan

		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;
		}
			menjalankan program.
			Pertama, pointer head diinisialisasi dengan nilai NULL, menandakan bahwa linked list kosong.
			Kemudian, tiga node baru dengan nilai data 6, 5, dan 2 dimasukkan ke awal linked list menggunakan fungsi push.
			Selanjutnya, node baru dengan nilai data 5 dimasukkan setelah node kedua (dengan nilai data 5) menggunakan fungsi insertAfter.
			Akhirnya, isi dari linked list dicetak menggunakan fungsi printList.

4	11	void push(struct Node** head_ref, int new_data)	Fungsi push:Fungsi ini beroperasi pada doubly linked list.Parameternya adalah:head_ref: Pointer ke pointer yang menunjuk ke head (awal) dari doubly linked list.new_data: Sebuah bilangan bulat yang mewakili nilai data dari node baru yang akan dimasukkan.

4	51-65	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:Fungsi ini beroperasi pada doubly linked list.Parameternya adalah:node: Pointer ke struct Node, yang merupakan head (awal) dari doubly linked list.

1	41-53	int main() {	Pertama, sebuah pointer head dari tipe struct Node*

		/* 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;
		}
			dideklarasikan dan diinisialisasi sebagai NULL. Ini menandakan bahwa linked list masih kosong saat awalnya.
			Kemudian, beberapa operasi dilakukan untuk mengubah linked list:
			Panggilan fungsi push digunakan untuk menambahkan node dengan nilai 6 di awal 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.