고급 C프로그래밍 High Level C Programming

CHAPTER 12

단순 연결 리스트

실습 1

문제 1)

단순 연결 리스트와 메서드들을 구현하시오.

(메서드 설명은 다음 페이지에)

C:\Users\Oculus\source\repos\ConsoleApplication2\x64\

```
*****
0. Create
1. display
2. Insert Node at beginning
3. Insert Node in specific position
4. Insert Node at end of LinkedList
5. Delete Node at beginning
6. Delete Node at end
7. Delete Node at position
8. ** To exit **
 Enter your choice: 0
Enter node data: 12

*****
0. Create
1. display
2. Insert Node at beginning
3. Insert Node in specific position
4. Insert Node at end of LinkedList
5. Delete Node at beginning
6. Delete Node at end
7. Delete Node at position
8. ** To exit **
 Enter your choice: 1
LinkedList: 12
```

```
*****
0. Create
1. display
2. Insert Node at beginning
3. Insert Node in specific position
4. Insert Node at end of LinkedList
5. Delete Node at beginning
6. Delete Node at end
7. Delete Node at position
8. ** To exit **
 Enter your choice: 2
Enter node data: 32

*****
0. Create
1. display
2. Insert Node at beginning
3. Insert Node in specific position
4. Insert Node at end of LinkedList
5. Delete Node at beginning
6. Delete Node at end
7. Delete Node at position
8. ** To exit **
 Enter your choice: 1
LinkedList: 32 12
```

```
*****
0. Create
1. display
2. Insert Node at beginning
3. Insert Node in specific position
4. Insert Node at end of LinkedList
5. Delete Node at beginning
6. Delete Node at end
7. Delete Node at position
8. ** To exit **
 Enter your choice: 3
Enter node data: 2
Enter position: 2

*****
0. Create
1. display
2. Insert Node at beginning
3. Insert Node in specific position
4. Insert Node at end of LinkedList
5. Delete Node at beginning
6. Delete Node at end
7. Delete Node at position
8. ** To exit **
 Enter your choice: 1
LinkedList: 32 1 2 12

*****
```

실습 1

정답

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
void create(); // 입력받은 값을 가진 linked list 생성
void display(); // 현재 저장된 linked list 출력
void insert_begin(); // head에 node 추가
void insert_end(); // tail에 node 추가
void insert_pos(); // 입력받은 위치에 node 추가
void delete_begin(); // head 삭제
void delete_end(); // tail 삭제
void delete_pos(); // 입력받은 위치의 node 삭제

struct node* head = NULL;

struct node
{
	int data;
	struct node* next;
};
```

실습 1

정답

```
int main()
{
        int choice;
        while (1)
        {
                printf("₩n*****₩n");
                printf("0. Create₩n");
                printf("1. display₩n");
                printf("2. Insert Node at beginning₩n");
                printf("3. Insert Node in specific position₩n");
                printf("4. Insert Node at end of LinkedList₩n");
                printf("5. Delete Node at beginning₩n");
                printf("6. Delete Node at end₩n");
                printf("7. Delete Node at position₩n");
                printf("8. ** To exit **");

                printf("₩n Enter your choice: ");
                scanf("%d", &choice);
                switch (choice)
                {
                case 0: create();
                        break;
                case 1: display();
                        break;
                case 2: insert_begin();
                        break;
                case 3: insert_pos();
                        break;
                case 4: insert_end();
                        break;
                case 5: delete_begin();
                        break;
                case 6: delete_end();
                        break;
                case 7: delete_pos();
                        break;
                case 8: exit(0);
                default:printf("₩n Wrong Choice");
                        break;
                }
        }
}
//creates a node
void create()
{
        struct node* temp;
        //creating new node
        temp = (struct node*)malloc(sizeof(struct node));
        printf("Enter node data: ");
        scanf("%d", &temp->data);
        temp->next = NULL;
        if (head == NULL) {
                head = temp;
        }
        else {
                struct node* ptr = head;
                while (ptr->next != NULL)
                {
                        ptr = ptr->next;
                }
                ptr->next = temp; //inserting at end of List
        }
}
```

실습 1

정답

```
// prints the entire LinkedList
void display()
{
        if (head == NULL)
        {
                printf("Linked List is Empty₩n");
                return;
        }
        printf("LinkedList: ");
        struct node* ptr = head;
        while (ptr != NULL) // start from first node
        {
                printf("%d ", ptr->data);
                ptr = ptr->next;
        }
        printf("₩n");
}
// to insert node at start of LinkedList
void insert_begin()
{
        struct node* temp;
        // creating a new node
        temp = (struct node*)malloc(sizeof(struct node));
        printf("Enter node data: ");
        scanf("%d", &temp->data);
        temp->next = NULL;
        if (head == NULL)
        {
                head = temp;
                return;
        }
        else
        {
                temp->next = head; //point it to old head node
                head = temp; //point head to new first node
        }
}
,
// to insert node at given position
void insert_pos()
{
        struct node* temp;
        // creating a new node
        temp = (struct node*)malloc(sizeof(struct node));
        printf("Enter node data: ");
        scanf("%d", &temp->data);
        temp->next = NULL;
        if (head == NULL) // if list empty we return
        {
                head = temp;
                return;
        }
        else
        {
                struct node* prev_ptr = NULL;
                struct node* ptr = head;
                int pos;
                printf("Enter position: ");
                scanf("%d", &pos);
                for (int i = 0; i < pos; i++)
                {
                        prev_ptr = ptr;
                        ptr = ptr->next;
                }
                //new node pointing to node in that pos
                temp->next = ptr;
                //prevptr pointing to new node
                prev_ptr->next = temp;
        }
}
```

실습 1

정답

```
// to insert node at end of LinkedList
void insert_end()
{
        struct node* temp;
        //creating new node
        temp = (struct node*)malloc(sizeof(struct node));
        printf("Enter node data: ");
        scanf("%d", &temp->data);
        temp->next = NULL;
        if (head == NULL)
        {
                head = temp; //if list is empty, we return
                return;
        }
        else {
                struct node* ptr = head;
                while (ptr->next != NULL)
                {
                        ptr = ptr->next;
                }
                // tail node pointing to new node
                ptr->next = temp;
        }
}
// to delete first node of LinkedList
void delete_begin()
{
        if (head == NULL) //if List is empty we return
        {
                printf("Linked List is empty | Nothing to delete ₩n");
                return;
        }
        else
        {
                struct node* ptr = head;
                head = head->next; // head node pointing to second node
                free(ptr); // deleting prev head node
                printf("Node Deleted ₩n");
        }
}
```

```
void delete_end()
{
        if (head == NULL) //if List is empty we return
        {
                printf("Linked List is empty | Nothing to delete ₩n");
                return;
        }
        else if (head->next == NULL)
        {
                struct node* ptr = head;
                head = ptr->next;
                free(ptr);
        }
        else
        {
                struct node* ptr = head;
                struct node* prev_ptr = NULL;
                while (ptr->next != NULL)// traverse till last but one node
                {
                        prev_ptr = ptr;
                        ptr = ptr->next;
                }
                prev_ptr->next = NULL; // next field of last but one field is made
as NULL
                free(ptr); // deleting last node
        }
}
```

실습 1

정답

```
{
// to delete node at given position
void delete_pos()
{
        int pos;
        printf("Enter node position to delete: ");
        scanf("%d", &pos);
        struct node* ptr = head;
        if (head == NULL) //we return if List is empty
        {
                printf("Linked List is empty ₩n");
                return;
        }
        else if (pos == 0)
        {
                ptr = head;
                head = ptr->next;  // head pointing to second node
                free(ptr); // deleting old first node
        }
        else
        {
                struct node* prev_ptr = NULL;
                for (int i = 0; i < pos; i++)
                {
                        prev_ptr = ptr;
                        ptr = ptr->next;
                }
                prev_ptr->next = ptr->next;  //prev node pointing to pos+1 node
                free(ptr); //deleting node at pos
        }
}
```

실습 1

문제 1) 메서드 설명

createLinkedList() // data가 null인 head를 가진 연결 리스트 생성

display() // 현재 연결 리스트에 연결된 node들의 data 출력

insert_begin() // 연결 리스트의 head에 node 삽입

insert_end() // 연결 리스트의 tail에 node 삽입

insert_pos() // position을 입력받고 원하는 position에 node 삽입

delete_begin() // 연결 리스트의 head 삭제

delete_end() // 연결 리스트의 tail 삭제

delete_pos() // position을 입력받고 원하는 position의 node 삭제

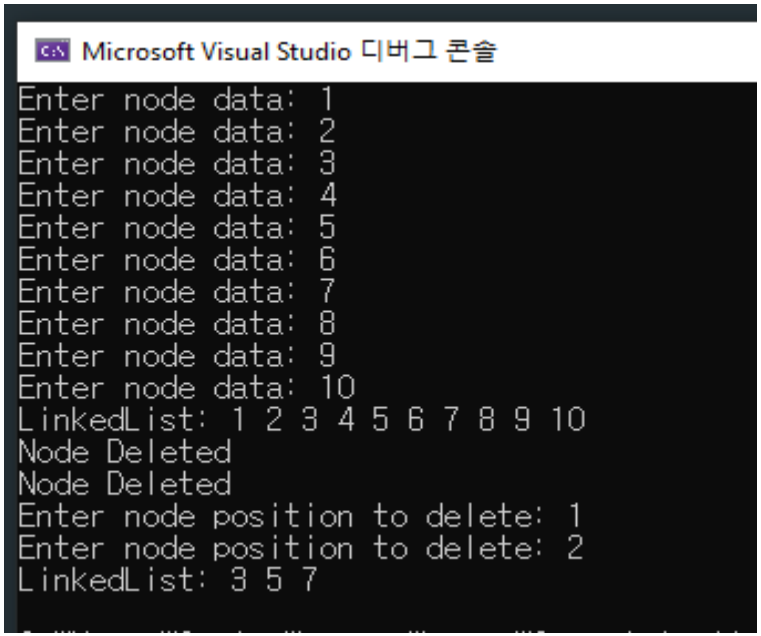
실습 2

문제 2)

문제 1에서 만든 메서드를 활용하여 다음과 같은 리스트를 만드시오.

[1,2,3,4,5,6,7,8,9,10] 의 값을 가지는 연결 리스트를 생성하고

[3,5,7] 의 값을 가지도록 메서드를 호출하시오. (호출 순서는 상관없음)



실습 2

정답

```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
void create(); // 입력받은 값을 가진 linked list 생성
void display(); // 현재 저장된 linked list 출력
void insert_begin(); // head에 node 추가
void insert_end(); // tail에 node 추가
void insert_pos(); // 입력받은 위치에 node 추가
void delete_begin(); // head 삭제
void delete_end(); // tail 삭제
void delete_pos(); // 입력받은 위치의 node 삭제

struct node* head = NULL;

struct node
{
	int data;
	struct node* next;
};

int main()
{
	create();
	for (int i = 0; i < 9; i++)
		insert_end();
	display();

	delete_begin();
	delete_begin();
	delete_end();
	delete_end();
	delete_end();
	delete_pos(); // 1 입력
	delete_pos(); // 2 입력

	display();
}
```

```c
//creates a node
void create()
{
	struct node* temp;
	//creating new node
	temp = (struct node*)malloc(sizeof(struct node));
	printf("Enter node data: ");
	scanf("%d", &temp->data);
	temp->next = NULL;
	if (head == NULL) {
		head = temp;
	}
	else {
		struct node* ptr = head;
		while (ptr->next != NULL)
		{
			ptr = ptr->next;
		}
		ptr->next = temp; //inserting at end of List
	}
}
// prints the entire LinkedList
void display()
{
	if (head == NULL)
	{
		printf("Linked List is Empty₩n");
		return;
	}
	printf("LinkedList: ");
	struct node* ptr = head;
	while (ptr != NULL) // start from first node
	{
		printf("%d ", ptr->data);
		ptr = ptr->next;
	}
	printf("₩n");
}
```

실습 2

정답

```
// to insert node at start of LinkedList
void insert_begin()
{
        struct node* temp;
        // creating  a new node
        temp = (struct node*)malloc(sizeof(struct node));
        printf("Enter node data: ");
        scanf("%d", &temp->data);
        temp->next = NULL;
        if (head == NULL)
        {
                head = temp;
                return;
        }
        else
        {
                temp->next = head; //point it to old head node
                head = temp; //point head to new first node
        }
}
```

```
// to insert node at given position
void insert_pos()
{
        struct node* temp;
        // creating  a new node
        temp = (struct node*)malloc(sizeof(struct node));
        printf("Enter node data: ");
        scanf("%d", &temp->data);
        temp->next = NULL;
        if (head == NULL) // if list empty we return
        {
                head = temp;
                return;
        }
        else
        {
                struct node* prev_ptr = NULL;
                struct node* ptr = head;
                int pos;
                printf("Enter position: ");
                scanf("%d", &pos);
                for (int i = 0; i < pos; i++)
                {
                        prev_ptr = ptr;
                        ptr = ptr->next;
                }
                //new node pointing to node in that pos
                temp->next = ptr;
                //prevptr pointing to new node
                prev_ptr->next = temp;
        }
}
```

실습 2

정답

```
// to insert node at end of LinkedList
void insert_end()
{
        struct node* temp;
        //creating new node
        temp = (struct node*)malloc(sizeof(struct node));
        printf("Enter node data: ");
        scanf("%d", &temp->data);
        temp->next = NULL;
        if (head == NULL)
        {
                head = temp; //if list is empty, we return
                return;
        }
        else {
                struct node* ptr = head;
                while (ptr->next != NULL)
                {
                        ptr = ptr->next;
                }
                // tail node pointing to new node
                ptr->next = temp;
        }
}
```

```
// to delete first node of LinkedList
void delete_begin()
{
        if (head == NULL) //if List is empty we return
        {
                printf("Linked List is empty | Nothing to delete ₩n");
                return;
        }
        else
        {
                struct node* ptr = head;
                head = head->next; // head node pointing to second node
                free(ptr); // deleting prev head node
                printf("Node Deleted ₩n");
        }
}
// to delete last node of LinkedList
void delete_end()
{
        if (head == NULL) //if List is empty we return
        {
                printf("Linked List is empty | Nothing to delete ₩n");
                return;
        }
        else if (head->next == NULL)
        {
                struct node* ptr = head;
                head = ptr->next;
                free(ptr);
        }
        else
        {
                struct node* ptr = head;
                struct node* prev_ptr = NULL;
                while (ptr->next != NULL)// traverse till last but one node
                {
                        prev_ptr = ptr;
                        ptr = ptr->next;
                }
                prev_ptr->next = NULL; // next field of last but one field is made
as NULL
                free(ptr); // deleting last node
        }
}
```

실습 2

정답

```
// to delete node at given position
void delete_pos()
{
        int pos;
        printf("Enter node position to delete: ");
        scanf("%d", &pos);
        struct node* ptr = head;
        if (head == NULL) //we return if List is empty
        {
                printf("Linked List is empty ₩n");
                return;
        }
        else if (pos == 0)
        {
                ptr = head;
                head = ptr->next;  // head pointing to second node
                free(ptr); // deleting old first node
        }
        else
        {
                struct node* prev_ptr = NULL;
                for (int i = 0; i < pos; i++)
                {
                        prev_ptr = ptr;
                        ptr = ptr->next;
                }
                prev_ptr->next = ptr->next;  //prev node pointing to pos+1 node
                free(ptr); //deleting node at pos
        }
}
```

실습 3

문제 3)

문제 1에서 만든 메서드를 활용하여 다음과 같은 리스트를 만드시오.

[1,2,3], [7,8] 의 값을 가지는 두 개의 연결 리스트를 생성하고

[1,2,3,7,8] 의 값을 가지도록 두 개의 연결 리스트를 연결하는 함수

concatList() 를 구현하시오.

Microsoft Visual Studio 디버그 콘솔

```
Create First LinkedList
Enter node data: 1
Enter node data: 2
Enter node data: 3
Create Second LinkedList
Enter the fisrt data: 9
Enter the second data: 10
LinkedList: 1 2 3 9 10
```

실습 3

정답

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
void create(); // 입력받은 값을 가진 linked list 생성
void display(); // 현재 저장된 linked list 출력
void insert_begin(); // head에 node 추가
void insert_end(); // tail에 node 추가
void insert_pos(); // 입력받은 위치에 node 추가
void delete_begin(); // head 삭제
void delete_end(); // tail 삭제
void delete_pos(); // 입력받은 위치의 node 삭제
void concatList();

struct node* head = NULL;
struct node* head2 = NULL;
struct node
{
	int data;
	struct node* next;
};
```

```
int main()
{
	printf("Create First LinkedList₩n");
	create();
	insert_end();
	insert_end();

	printf("Create Second LinkedList₩n");
	struct node* temp = (struct node*)malloc(sizeof(struct node));
	head2 = temp;
	printf("Enter the fisrt data: ");
	scanf("%d", &head2->data);
	struct node* ptr = (struct node*)malloc(sizeof(struct node));
	temp->next = ptr;
	printf("Enter the second data: ");
	scanf("%d", &ptr->data);
	ptr->next = NULL;
	concatList();
	display();

	free(head);
	free(head2);
	free(ptr);
}
```

실습 3

정답

```
void concatList() {
        struct node* temp = head2;
        //creating new node
        if (head == NULL)
        {
                head = temp; //if list is empty, we return
                return;
        }
        else {
                struct node* ptr = head;
                while (ptr->next != NULL)
                {
                        ptr = ptr->next;
                }
                // tail node pointing to new node
                ptr->next = temp;
        }
}
//creates a node
void create()
{
        struct node* temp;
        //creating new node
        temp = (struct node*)malloc(sizeof(struct node));
        printf("Enter node data: ");
        scanf("%d", &temp->data);
        temp->next = NULL;
        if (head == NULL) {
                head = temp;
        }
        else {
                struct node* ptr = head;
                while (ptr->next != NULL)
                {
                        ptr = ptr->next;
                }
                ptr->next = temp; //inserting at end of List
        }
}
// prints the entire LinkedList
void display()
{
        if (head == NULL)
        {
                printf("Linked List is Empty₩n");
                return;
        }
        printf("LinkedList: ");
        struct node* ptr = head;
        while (ptr != NULL) // start from first node
        {
                printf("%d ", ptr->data);
                ptr = ptr->next;
        }
        printf("₩n");
}
// to insert node at start of LinkedList
void insert_begin()
{
        struct node* temp;
        // creating a new node
        temp = (struct node*)malloc(sizeof(struct node));
        printf("Enter node data: ");
        scanf("%d", &temp->data);
        temp->next = NULL;
        if (head == NULL)
        {
                head = temp;
                return;
        }
        else
        {
                temp->next = head; //point it to old head node
                head = temp; //point head to new first node
        }
}
```

실습 3

정답

```
// to insert node at given position
void insert_pos()
{
        struct node* temp;
        // creating a new node
        temp = (struct node*)malloc(sizeof(struct node));
        printf("Enter node data: ");
        scanf("%d", &temp->data);
        temp->next = NULL;
        if (head == NULL) // if list empty we return
        {
                head = temp;
                return;
        }
        else
        {
                struct node* prev_ptr = NULL;
                struct node* ptr = head;
                int pos;
                printf("Enter position: ");
                scanf("%d", &pos);
                for (int i = 0; i < pos; i++)
                {
                        prev_ptr = ptr;
                        ptr = ptr->next;
                }
                //new node pointing to node in that pos
                temp->next = ptr;
                //prevptr pointing to new node
                prev_ptr->next = temp;
        }
}
```

```
// to insert node at end of LinkedList
void insert_end()
{
        struct node* temp;
        //creating new node
        temp = (struct node*)malloc(sizeof(struct node));
        printf("Enter node data: ");
        scanf("%d", &temp->data);
        temp->next = NULL;
        if (head == NULL)
        {
                head = temp; //if list is empty, we return
                return;
        }
        else {
                struct node* ptr = head;
                while (ptr->next != NULL)
                {
                        ptr = ptr->next;
                }
                // tail node pointing to new node
                ptr->next = temp;
        }
}
// to delete first node of LinkedList
void delete_begin()
{
        if (head == NULL) //If List is empty we return
        {
                printf("Linked List is empty | Nothing to delete ₩n");
                return;
        }
        else
        {
                struct node* ptr = head;
                head = head->next; // head node pointing to second node
                free(ptr); // deleting prev head node
                printf("Node Deleted ₩n");
        }
}
```

실습 3

정답

```
// to delete last node of LinkedList
void delete_end()
{
        if (head == NULL) //if List is empty we return
        {
                printf("Linked List is empty | Nothing to delete ₩n");
                return;
        }
        else if (head->next == NULL)
        {
                struct node* ptr = head;
                head = ptr->next;
                free(ptr);
        }
        else
        {
                struct node* ptr = head;
                struct node* prev_ptr = NULL;
                while (ptr->next != NULL)// traverse till last but one node
                {
                        prev_ptr = ptr;
                        ptr = ptr->next;
                }
                prev_ptr->next = NULL; // next field of last but one field is made
as NULL
                free(ptr); // deleting last node
        }
}
```

```
// to delete node at given position
void delete_pos()
{
        int pos;
        printf("Enter node position to delete: ");
        scanf("%d", &pos);
        struct node* ptr = head;
        if (head == NULL) //we return if List is empty
        {
                printf("Linked List is empty ₩n");
                return;
        }
        else if (pos == 0)
        {
                ptr = head;
                head = ptr->next; // head pointing to second node
                free(ptr); // deleting old first node
        }
        else
        {
                struct node* prev_ptr = NULL;
                for (int i = 0; i < pos; i++)
                {
                        prev_ptr = ptr;
                        ptr = ptr->next;
                }
                prev_ptr->next = ptr->next; //prev node pointing to pos+1 node
                free(ptr); //deleting node at pos
        }
}
```

실습 4

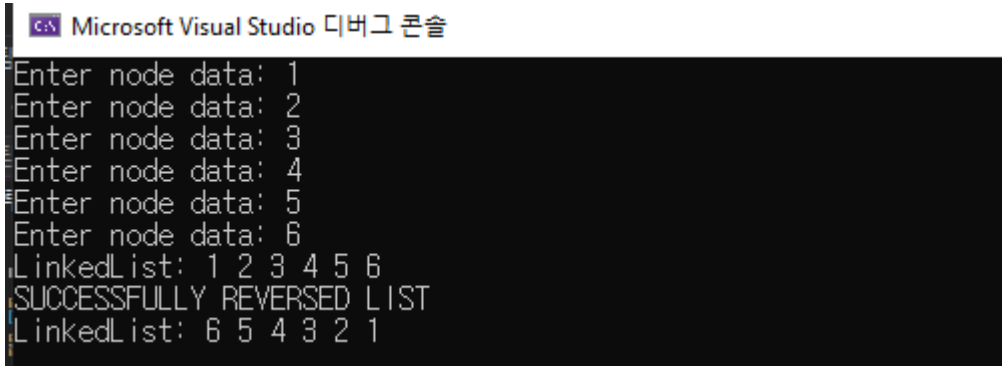
문제 4)

문제 1에서 만든 메서드를 활용하여 다음과 같은 리스트를 만드시오.

[1,2,3,4,5,6] 의 값을 가지는 연결 리스트를 생성하고

[6,5,4,3,2,1] 의 값을 가지도록 바꾸는

reverseLinkedList() 함수를 구현하시오.



실습 4

정답

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
void create(); // 입력받은 값을 가진 linked list 생성
void display(); // 현재 저장된 linked list 출력
void insert_begin(); // head에 node 추가
void insert_end(); // tail에 node 추가
void insert_pos(); // 입력받은 위치에 node 추가
void delete_begin(); // head 삭제
void delete_end(); // tail 삭제
void delete_pos(); // 입력받은 위치의 node 삭제
void reverseLinkedList();
struct node* head = NULL;

struct node
{
	int data;
	struct node* next;
};
```

```
int main()
{
	create();
	insert_end();
	insert_end();
	insert_end();
	insert_end();
	insert_end();
	display();

	reverseLinkedList();
	display();
}

void reverseLinkedList() {

	struct node* prevNode, * curNode;

	if (head != NULL)
	{
		prevNode = head;
		curNode = head->next;
		head = head->next;

		prevNode->next = NULL; // Make first node as last node

		while (head != NULL)
		{
			head = head->next;
			curNode->next = prevNode;

			prevNode = curNode;
			curNode = head;
		}

		head = prevNode; // Make last node as head

		printf("SUCCESSFULLY REVERSED LIST₩n");
	}

}
```

실습 4

정답

```
void create()
{
        struct node* temp;
        //creating new node
        temp = (struct node*)malloc(sizeof(struct node));
        printf("Enter node data: ");
        scanf("%d", &temp->data);
        temp->next = NULL;
        if (head == NULL) {
                head = temp;
        }
        else {
                struct node* ptr = head;
                while (ptr->next != NULL)
                {
                        ptr = ptr->next;
                }
                ptr->next = temp; //inserting at end of List
        }
}
// prints the entire LinkedList
void display()
{
        if (head == NULL)
        {
                printf("Linked List is Empty₩n");
                return;
        }
        printf("LinkedList: ");
        struct node* ptr = head;
        while (ptr != NULL) // start from first node
        {
                printf("%d ", ptr->data);
                ptr = ptr->next;
        }
        printf("₩n");
}
```

```
// to insert node at start of LinkedList
void insert_begin()
{
        struct node* temp;
        // creating a new node
        temp = (struct node*)malloc(sizeof(struct node));
        printf("Enter node data: ");
        scanf("%d", &temp->data);
        temp->next = NULL;
        if (head == NULL)
        {
                head = temp;
                return;
        }
        else
        {
                temp->next = head; //point it to old head node
                head = temp; //point head to new first node
        }
}
```

```
// to insert node at given position
void insert_pos()
{
        struct node* temp;
        // creating a new node
        temp = (struct node*)malloc(sizeof(struct node));
        printf("Enter node data: ");
        scanf("%d", &temp->data);
        temp->next = NULL;
        if (head == NULL) // if list empty we return
        {
                head = temp;
                return;
        }
        else
        {
                struct node* prev_ptr = NULL;
                struct node* ptr = head;
                int pos;
                printf("Enter position: ");
                scanf("%d", &pos);
                for (int i = 0; i < pos; i++)
                {
                        prev_ptr = ptr;
                        ptr = ptr->next;
                }
                //new node pointing to node in that pos
                temp->next = ptr;
                //prevptr pointing to new node
                prev_ptr->next = temp;
        }
}
```

실습 4

정답

```
// to insert node at end of LinkedList
void insert_end()
{
        struct node* temp;
        //creating new node
        temp = (struct node*)malloc(sizec
        printf("Enter node data: ");
        scanf("%d", &temp->data);
        temp->next = NULL;
        if (head == NULL)
        {
                head = temp; //if list is emp
                return;
        }
        else {
                struct node* ptr = head;
                while (ptr->next != NULL)
                {
                        ptr = ptr->next;
                }
                // tail node pointing to new
                ptr->next = temp;
        }
}
// to delete first node of LinkedList
void delete_begin()
{
        if (head == NULL) //if List is empty
        {
                printf("Linked List is empty |
                return;
        }
        else
        {
                struct node* ptr = head;
                head = head->next; // hea
                free(ptr); // deleting prev head node
                printf("Node Deleted ₩n");
        }
}
```

```
// to delete last node of LinkedList
void delete_end()
{
        if (head == NULL) //if List is empty we return
        {
                printf("Linked List is empty | Nothing to delete ₩n");
                return;
}
// to delete node at given position
void delete_pos()
{
        int pos;
        printf("Enter node position to delete: ");
        scanf("%d", &pos);
        struct node* ptr = head;
        if (head == NULL) //we return if List is empty
        {
                printf("Linked List is empty ₩n");
                return;
        }
        else if (pos == 0)
        {
                ptr = head;
                head = ptr->next; // head pointing to second node
                free(ptr); // deleting old first node
        }
        else
        {
                struct node* prev_ptr = NULL;
                for (int i = 0; i < pos; i++)
                {
                        prev_ptr = ptr;
                        ptr = ptr->next;
                }
                prev_ptr->next = ptr->next; //prev node pointing to pos+1 node
                free(ptr); //deleting node at pos
        }
}
```

ut one node

ut one field is made

실습 4

정답

```
}
// to delete node at given position
void delete_pos()
{
        int pos;
        printf("Enter node position to delete: ");
        scanf("%d", &pos);
        struct node* ptr = head;
        if (head == NULL) //we return if List is empty
        {
                printf("Linked List is empty ₩n");
                return;
        }
        else if (pos == 0)
        {
                ptr = head;
                head = ptr->next;  // head pointing to second node
                free(ptr); // deleting old first node
        }
        else
        {
                struct node* prev_ptr = NULL;
                for (int i = 0; i < pos; i++)
                {
                        prev_ptr = ptr;
                        ptr = ptr->next;
                }
                prev_ptr->next = ptr->next;  //prev node pointing to pos+1 node
                free(ptr); //deleting node at pos
        }
}
```

실습 5

문제 5)

연결 리스트에 저장된 값이 Palindrome 인지 검사하는

isPalindrome() 함수를 구현하시오.

Palindrome: 거꾸로 읽어도 제대로 읽는 것과 같은 문장이나 낱말, 숫자, 문자열 등

Microsoft Visual Studio 디버그 콘솔

```
Create LinkedList
Enter node data: 1
Enter node data: 2
Enter node data: 3
Enter node data: 4
LinkedList: 1 2 3 4
The linked list is not a palindrome.
```

Microsoft Visual Studio 디버그 콘솔

```
Create LinkedList
Enter node data: 1
Enter node data: 2
Enter node data: 2
Enter node data: 1
LinkedList: 1 2 2 1
The linked list is a palindrome.
```

실습 5

정답

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
void create(); // 입력받은 값을 가진 linked list 생성
void display(); // 현재 저장된 linked list 출력
void insert_begin(); // head에 node 추가
void insert_end(); // tail에 node 추가
void insert_pos(); // 입력받은 위치에 node 추가
void delete_begin(); // head 삭제
void delete_end(); // tail 삭제
void delete_pos(); // 입력받은 위치의 node 삭제
int isPalindrome();

struct node* head = NULL;

struct node
{
	int data;
	struct node* next;
};

int counter = 0; // node 개수
int main()
{
	// 4개의 node로 이루어진 linkedlist 생성
	printf("Create LinkedList₩n");
	create();
	counter++;
	insert_end();
	counter++;
	insert_end();
	counter++;
	insert_end();
	counter++;
	// linked list에 저장된 값 출력
	display();

	// palindrome인지 아닌지 결과 값 저장
	int result;
	result = isPalindrome();
	if (result == 1)
	{
		printf("The linked list is a palindrome.₩n");
	}
	else
	{
		printf("The linked list is not a palindrome.₩n");
	}
	return 0;
}
```

실습 5

```
int isPalindrome() {
	int i = 0, j;
	struct node* front, * rear;

	while (i != counter / 2) // 반반 나눠서 앞뒤로 중간지점부터 양 끝쪽으로 퍼
지면서 check
	{
		front = rear = head;
		for (j = 0; j < i; j++)
		{
			front = front->next;
		}
		for (j = 0; j < counter - (i + 1); j++)
		{
			rear = rear->next;
		}
		if (front->data != rear->data)
		{
			return 0;
		}
		else
		{
			i++;
		}
	}

	return 1;
}
```

```
//creates a node
void create()
{
	struct node* temp;
	//creating new node
	temp = (struct node*)malloc(sizeof(struct node));
	printf("Enter node data: ");
	scanf("%d", &temp->data);
	temp->next = NULL;
	if (head == NULL) {
		head = temp;
	}
	else {
		struct node* ptr = head;
		while (ptr->next != NULL)
		{
			ptr = ptr->next;
		}
		ptr->next = temp; //inserting at end of List
	}
}
```

실습 5

정답

```
// prints the entire LinkedList
void display()
{
        if (head == NULL)
        {
                printf("Linked List is Empty₩n");
                return;
        }
        printf("LinkedList: ");
        struct node* ptr = head;
        while (ptr != NULL) // start from first node
        {
                printf("%d ", ptr->data);
                ptr = ptr->next;
        }
        printf("₩n");
}
// to insert node at start of LinkedList
void insert_begin()
{
        struct node* temp;
        // creating a new node
        temp = (struct node*)malloc(sizeof(struct node));
        printf("Enter node data: ");
        scanf("%d", &temp->data);
        temp->next = NULL;
        if (head == NULL)
        {
                head = temp;
                return;
        }
        else
        {
                temp->next = head; //point it to old head node
                head = temp; //point head to new first node
        }
}
```

```
// to insert node at given position
void insert_pos()
{
        struct node* temp;
        // creating a new node
        temp = (struct node*)malloc(sizeof(struct node));
        printf("Enter node data: ");
        scanf("%d", &temp->data);
        temp->next = NULL;
        if (head == NULL) // if list empty we return
        {
                head = temp;
                return;
        }
        else
        {
                struct node* prev_ptr = NULL;
                struct node* ptr = head;
                int pos;
                printf("Enter position: ");
                scanf("%d", &pos);
                for (int i = 0; i < pos; i++)
                {
                        prev_ptr = ptr;
                        ptr = ptr->next;
                }
                //new node pointing to node in that pos
                temp->next = ptr;
                //prevptr pointing to new node
                prev_ptr->next = temp;
        }
}
```

실습 5

정답

```
// to insert node at end of LinkedList
void insert_end()
{
        struct node* temp;
        //creating new node
        temp = (struct node*)malloc(sizeof(struct node));
        printf("Enter node data: ");
        scanf("%d", &temp->data);
        temp->next = NULL;
        if (head == NULL)
        {
                head = temp; //if list is empty, we return
                return;
        }
        else {
                struct node* ptr = head;
                while (ptr->next != NULL)
                {
                        ptr = ptr->next;
                }
                // tail node pointing to new node
                ptr->next = temp;
        }
}
```

```
// to delete first node of LinkedList
void delete_begin()
{
        if (head == NULL) //if List is empty we return
        {
                printf("Linked List is empty | Nothing to delete ₩n");
                return;
        }
        else
        {
                struct node* ptr = head;
                head = head->next; // head node pointing to second node
                free(ptr); // deleting prev head node
                printf("Node Deleted ₩n");
        }
}
```

```
// to delete last node of LinkedList
void delete_end()
{
        if (head == NULL) //if List is empty we return
        {
                printf("Linked List is empty | Nothing to delete ₩n");
                return;
        }
        else if (head->next == NULL)
        {
                struct node* ptr = head;
                head = ptr->next;
                free(ptr);
        }
        else
        {
                struct node* ptr = head;
                struct node* prev_ptr = NULL;
                while (ptr->next != NULL)// traverse till last but one node
                {
                        prev_ptr = ptr;
                        ptr = ptr->next;
                }
                prev_ptr->next = NULL; // next field of last but one field is made
as NULL
                free(ptr); // deleting last node
        }
}
```

실습 5

정답

```
,
// to delete node at given position
void delete_pos()
{
        int pos;
        printf("Enter node position to delete: ");
        scanf("%d", &pos);
        struct node* ptr = head;
        if (head == NULL) //we return if List is empty
        {
                printf("Linked List is empty ₩n");
                return;
        }
        else if (pos == 0)
        {
                ptr = head;
                head = ptr->next;  // head pointing to second node
                free(ptr); // deleting old first node
        }
        else
        {
                struct node* prev_ptr = NULL;
                for (int i = 0; i < pos; i++)
                {
                        prev_ptr = ptr;
                        ptr = ptr->next;
                }
                prev_ptr->next = ptr->next;  //prev node pointing to pos+1 node
                free(ptr); //deleting node at pos
        }
}
```

질문 및 정리