4-23

1. 試撰寫出將一節點 new 加在 x 節點之後之片段程式

```
void insert_after(Node *x, Node *new) {
    new->next = x->next;
    x->next = new;
}
```

2. 試撰寫如何將 x 節點刪除之片段程式

```
void delete(Node *x) {
    Node *prev = head;
    while (prev->next != x) {
        prev = prev->next;
    }
    prev->next = x->next;
    free(x);
}
```

4-30 利用環狀串列來表示堆疊的加入和刪除

```c
#include <stdio.h>
#include <stdlib.h>

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

struct stack *top = NULL;

void push(int data) {
    struct stack *new = (struct stack *)malloc(sizeof(struct stack));
    new->data = data;
    new->next = top;
    top = new;
}

int pop() {
    if (top == NULL) {
        printf("Stack is empty\n");
        return -1;
    }
    struct stack *temp = top;
    int data = temp->data;
    top = top->next;
    free(temp);
    return data;
}
```

動動腦

3. 試比較分析單鏈結串列與雙鏈結串列有何優缺點。

單鏈結串列：

- 優點：

    - 不需要額外的內存來存儲左右指針。
- 缺點：
    - 查找特定元素時，需要從頭節點開始逐個遍歷，效率較低。
    - 只能單向遍歷，不能反向遍歷。

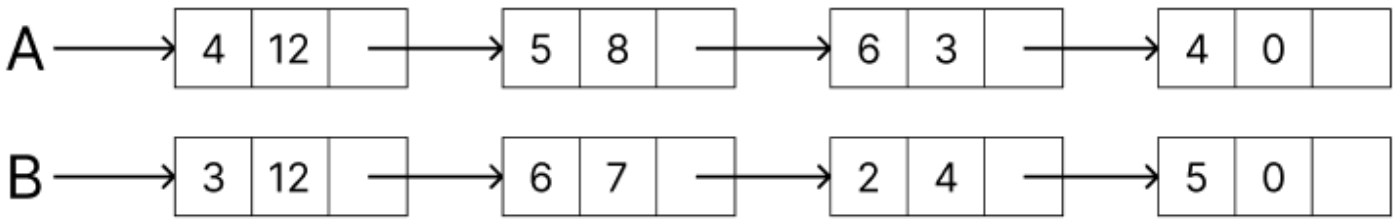
雙鏈結串列：

- 優點：
    - 可以雙向遍歷，查找特定元素時，可以從兩個方向進行，效率較高。
    - 可以方便地在任意位置插入和刪除節點。
- 缺點：
    - 需要更多的內存來存儲左右指針。
    - 插入和刪除節點時，需要修改更多的指針。

4. 請利用單向鏈結串列來表示兩個多項式，例如

$$A = 4x^{12} + 5x^8 + 6x^3 + 4$$

$$B = 3x^{12} + 6x^7 + 2x^4 + 5$$



8. 請撰寫雙向鏈結串列的加入和刪除前端結點的演算法，此處假設前端 head 節點有存放資料。

```c
#include <stdio.h>
#include <stdlib.h>

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

struct node *head = NULL;

void insert_front(int data) {
    struct node *new = (struct node *)malloc(sizeof(struct node));
    new->data = data;
    new->prev = NULL;
    new->next = head;
    if (head != NULL) {
        head->prev = new;
    }
    head = new;
}

void delete_front() {
    if (head == NULL) {
        printf("List is empty\n");
        return;
    }
    struct node *temp = head;
    head = head->next;
    if (head != NULL) {
        head->prev = NULL;
    }
    free(temp);
}
```

9. 請撰寫回收整個雙向鏈結串列之片段程式

```
void free_list() {
    struct node *temp;
    while (head != NULL) {
        temp = head;
        head = head->next;
        free(temp);
    }
}
```