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

struct A {
    char name[20];
    int age;
    int salary;
    struct A *next;
};

void input(struct A **, struct A **);
void output(struct A **, struct A **);
void find(struct A **, struct A **);
void rem(struct A **, struct A **);
void node_free(struct A **, struct A **);

int main(void)
{
    char ch;
    int cont = 1;

    struct A *head, *tail;
    head = tail = NULL;

    while(cont)
    {
        printf("₩n *** Menu *** ₩n");
        printf("1. Input a node ₩n");
        printf("2. Print all nodes ₩n");
        printf("3. Find a node ₩n");
        printf("4. Remove a node ₩n");
        printf("5. Free all nodes ₩n");
        printf("6. Exit ₩n");
        printf("₩n Choose the item : ");

        ch = getchar();
        while(getchar() != '₩n'); //fflush(stdin);

        switch(ch)
        {
            case '1' : input(&head, &tail);
                break;
            case '2' : output(&head, &tail);
                break;
            case '3' : find(&head, &tail);
                break;
            case '4' : rem(&head, &tail);
                break;
            case '5' : node_free(&head, &tail);
                break;
            case '6' : cont = 0;
                break;
            default : printf("Wrong input.");
        }
    }
    return 0;
}

void input(struct A **head, struct A **tail) //노드 추가
{
    struct A *ptr;

    while(1)
    {
```

```
        if((ptr = (struct A *)malloc(sizeof(struct A))) == NULL) //메모리 요청
        {
            printf("Memory Allocation Error ₩n");
            exit(1);
        }

        printf("₩n성명 (end to exit) ? ");
        gets(ptr->name);
        if(!strcmp(ptr->name, "end"))
            break;

        printf("나이 ? ");
        scanf("%d", &ptr->age);

        printf("월급 ? ");
        scanf("%d", &ptr->salary);

        while(getchar() != '₩n'); //fflush(stdin);

        ptr->next = NULL;

        if(*head == NULL)
            *head = *tail = ptr; // 첫번째 노드인 경우
        else
        {
            (*tail)->next = ptr; // 노드가 추가되는 경우
            *tail = ptr;
        }
    }

    free(ptr);
}

void output(struct A **head, struct A **tail) //전체 노드 출력
{
    struct A *ptr;

    printf("₩nNode List ₩n");
    ptr = *head;
    while(ptr)
    {
        printf("name:%s, age: %d, salary: %d ₩n",
        ptr->name, ptr->age, ptr->salary);
        ptr = ptr->next;
    }
}

void find(struct A **head, struct A **tail) // 노드 검색
{
    struct A *ptr;
    int t_salary, found = 0;

    printf("입력한 월급 이상의 노드 검색 ? ");
    scanf("%d", &t_salary);

    ptr = *head;
    while(ptr)
    {
        if(ptr->salary >= t_salary)
        { //참이면 출력
            printf("name:%s , age: %d, salary: %d ₩n",
            ptr->name, ptr->age, ptr->salary);
            found = 1;
        }
        ptr = ptr->next; //다음 노드 주소로 이동
```

```
    }
    while(getchar() != '₩n'); //fflush(stdin);

    if( !found )
        printf("Not found. ₩n");

}

void rem(struct A **head, struct A **tail) // 노드 삭제 및 해제
{
    struct A *ptr, *prev;
    int found = 0;
    char s_name[20], ch;

    printf("₩n삭제할 성명 ? ");
    gets(s_name);

    ptr = *head;

    while(ptr)
    {
        if( !strcmp(ptr->name, s_name) ) //성명비교
        {
            printf("name:%s , age: %d, salary: %d ₩n",
            ptr->name, ptr->age, ptr->salary);

            printf("출력된 노드를 삭제할까요 ? (y/n) ");
            ch = getchar();
            while(getchar() != '₩n'); //fflush(stdin);

            if(ch=='Y' || ch=='y')
            {
                if(ptr == *head)
                {
                    *head = ptr->next; //첫 번째 노드인 경우
                }
                else if(ptr == *tail) //마지막 노드인 경우
                {
                    prev->next = NULL;
                    *tail = prev;
                }
                else
                {
                    prev->next = ptr->next; //중간 노드인 경우
                }

                found = 1;
                free(ptr); //삭제된 노드 해제
                break; //삭제하였으므로 반복문 탈출
            }
            else
                break; //반복문 탈출
        } //if end

        prev = ptr;
        ptr = ptr->next; //다음 노드 시작주소로 이동

    } //while end

    if(found)
        printf("노드 삭제 완료. ₩n");
    else
        printf("Not found. ₩n");

    output(head, tail); // 삭제 되었는지 확인
```

```
}

void node_free(struct A **head, struct A **tail) // 전체 노드 삭제 및 해제
{
    struct A *ptr, *x;

    ptr = *head;
    if(ptr == NULL)
    {
        printf("삭제될 노드가 없습니다. ₩n");
        return;
    }

    printf("₩n모든 node가 메모리에서 제거됩니다. ₩n");
    while(ptr)
    {
        x = ptr;
        ptr = ptr->next; //다음 노드 주소로 이동
        free(x); //노드 영역 해제
    }

    *head = *tail = NULL;
    output(head, tail); //노드 List가 비어 있음을 확인한다.
}
```