Linked list 관련

○ 동적 메모리 할당 함수
동적 메모리 할당 함수는 다음 3가지이다.
void *calloc (size_t nmemb, size_t size);
void *malloc (size_t size);
void *realloc (void *ptr, size_t size);

이들은 운영체제로부터 동적으로 메모리를 할당받아 그 주소를 돌려주거나 실패하는 경우 NULL을 돌려준다.

○ 동적 메모리 해제 함수
free (void *ptr);

동적으로 할당 받은 메모리는 더 이상 필요가 없는 경우 반드시 해제하도록 한다. 해제하지 않는 경우, 예기치 않은 문제를 만들 수 있다.

```
struct lnk_list *ptr;

prt = malloc(sizeof(struct lnk_list));

...

free(ptr);
ptr = NULL;
```

○ linked list 앞부분에 추가

```
struct lnk_list * insert_front (struct lnk_list * original, int data)
{
        struct lnk_list   *temp;

        temp = malloc(sizeof (struct lnk_list));
        if (temp) {
                temp->data = data;
                if (original) {
                        temp -> next = original;
                }
                else {
                        temp -> next = NULL;
                }
        }
        else {
                fprintf(stderr, "Memory Full₩n");
        }

        return (temp);
}
```

○ linked list 맨 뒤에 추가

```
struct lnk_list * insert_rear (struct lnk_list * original, int data)
{
        struct lnk_list   *temp;
        strcut lnk_list   *ptr;

        temp = malloc(sizeof (struct lnk_list));
        if (temp) {
                temp->data = data;
                temp->next = NULL;

                if (original) {
                        for (ptr = original; ptr -> next; ptr = ptr -> next);
                        ptr -> next = temp;
                }
                else {
                        return (temp);
                }
        }
        else {
                fprintf(stderr, "Memory Full₩n");
        }
        return (original);
}
```

○ 순서에 맞도록 linked list에 추가

```
struct lnk_list * sorted_insert (struct lnk_list * original, int data)
{
        struct lnk_list   *temp;
        strcut lnk_list   *pre_ptr, *ptr;

        temp = malloc(sizeof (struct lnk_list));
        if (temp) {
                temp->data = data;

                if (original) {
                        for (ptr = original; ptr && (data < ptr -> data);
                                        pre_ptr = ptr, ptr = ptr -> next);
                        if (ptr == original) {     /* 맨 앞에 삽입 */
                                temp->next = original;
                                return (temp);
                        }
                        else if (!ptr) {           /* 맨 뒤에 삽입 */
                                pre_ptr -> next = temp;
                                temp->next = NULL;
                                return (original);
                        }
                        else {                     /* 중간에 삽입 */
                                pre_ptr -> next = temp;
                                temp -> next = ptr;
                                return (original);
                        }
                }
                else {                             /* 단독 */
                        temp -> next = NULL;
                        return (temp);
                }
        }
        else {
                fprintf(stderr, "Memory Full₩n");
        }
        return (original);
}
```