

박기홍 6일차 과제

1. HW_001

[소스코드]

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>

int main() {

    // 변수 선언하기.
    char* inputText = (char*)malloc(sizeof(char) * 100);
    char tempChar, mostChar; // 최다 등장 비교용도.
    int cmpCnt = 0, tempCmpCnt = 0;
    int i, j, textCnt = 0, textSpaceCnt = 0;

    // 입출력 받기.
    printf("입력 : ");
    scanf("%[^\n]s", inputText); // 공백으로 입력 받기 위한 연산자 (광운대 SW예비학교
    때 배운 내용)

    for (i = 0; *(inputText + i); i++) {
        if (*(inputText + i) != 32) { // ASCII Code값이 32 (공백)이 아니라면,
            textCnt++;
        }else {
            textSpaceCnt++;
        }
    }

    // 최다등장문자 구하는 알고리즘.
    for (i = 0; i < textCnt + textSpaceCnt; i++) {
        tempChar = *(inputText + i);
        if (*(inputText + i) == 32) { // 공백은 최다등장문자에 포함하지 않음.
            continue;
        }
        for (j = 0; j < textCnt + textSpaceCnt; j++) {
            // 만약 해당 문자가 입력된 문자열에 있다면,
            if (tempChar == *(inputText + j)) {
                tempCmpCnt++;
            }
        }
        // 문자열의 등장 횟수를 비교해서,
        // 해당 문자의 등장 횟수가 기존에 등장한 문자의 횟수보다 많다면,
        // 기록을 갱신함.
        if (tempCmpCnt > cmpCnt) {
            cmpCnt = tempCmpCnt;
            mostChar = tempChar;
        }
    }
}
```

```

        tempCmpCnt = 0;
    }

    // 형식 출력하기.
    printf("Wn출력 : ");
    // 입력 받은 문자열 거꾸로 출력하기.
    for (i = textCnt + textSpaceCnt; i > 0; i--) {
        printf("%c", *(inputText + (i - 1)));
    }
    // 최다등장문자 출력하기. (공백 제외)
    printf("Wn      최다등장문자 : %c", mostChar);

    // 동적할당 메모리 해제하기.
    free(inputText);

    return 0;
}

```

[실행결과]

Test Case #1

```

Microsoft Visual Studio 디버그
입력 : I like robot!
출력 : !tobor ekil I
      최다등장문자 : o
C:\Users\lordk\Github\ROBIT_Intern_KiHongPark_HW_repo\Projects\C_Language\Homework\HW_20240708\x64\Debu
(프로세스 20500개)이(가) 종료되었습니다(코드: 0개).
이 창을 닫으려면 아무 키나 누르세요...

```

Test Case #2

```
Microsoft Visual Studio 디버그 × + ▾
입력 : I wish take part in robit!

출력 : !tibor ni trap ekat hsiw I
      최다등장문자 : i
C:\Users\lordk\Github\ROBIT_Intern_KiHongPark_HW_repo\Projects\C_Language\HomeWork\HW_20240708\x64\Debug\HW_20240708.exe
(프로세스 13948개)이(가) 종료되었습니다(코드: 0개).
이 창을 닫으려면 아무 키나 누르세요...
```

Test Case #3

```
Microsoft Visual Studio 디버그 × + ▾
입력 : Hello, World!

출력 : !dlroW ,olleH
      최다등장문자 : l
C:\Users\lordk\Github\ROBIT_Intern_KiHongPark_HW_repo\Projects\C_Language\HomeWork\HW_20240708\x64\Debug\HW_20240708.exe
(프로세스 20300개)이(가) 종료되었습니다(코드: 0개).
이 창을 닫으려면 아무 키나 누르세요...
```

2. HW_002

[소스코드]

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>

int main() {

    // 변수 선언하기.
```

```

int numAry[3][4] = { 0, };
int* numList = (int*)calloc(12, sizeof(4)); // 입력되는 숫자 저장용(stack)
int i, j;
/*
 * idxCnt : 입력받은 숫자를 numList에 집어 넣기 위함.
 * idxCirculation : numAry에 numList의 요소를 순환하여 채워 넣기 위함.
 */
int inputNum, nextInput, idxCnt = 0, idxCirculation = 0;

// 형식 입출력 받기.
printf("input : ");
scanf("%d", &inputNum);
if (inputNum == 0) { // 입력 값이 0이면 프로그램 종료하기.
    printf("프로그램을 종료합니다.");
    return 0;
}
*(numList + idxCnt) = inputNum;

// (최초) 입력 받은 숫자로 배열 전체 지정하기.
for (i = 0; i < 3; i++) {
    for (j = 0; j < 4; j++) {
        numAry[i][j] = inputNum;
    }
}

// (최초) 입력 받은 숫자 출력하기.
// 입력 받은 숫자로 배열 전체 지정 및 출력하기.
printf("\n");
for (i = 0; i < 3; i++) {
    for (j = 0; j < 4; j++) {
        numAry[i][j] = *(numList + idxCirculation);
        if (*(numList + idxCirculation) == 0) {
            idxCirculation = 0;
            numAry[i][j] = *(numList + idxCirculation);
            idxCirculation++;
        } else {
            idxCirculation++;
        }
        printf("%d ", numAry[i][j]);
    }
    printf("\n");
}

idxCirculation = 0;

while (1){

    printf("\nnext input : ");
    scanf("%d", &nextInput);
    if (nextInput == 0) { // 입력 값이 0이면 프로그램 종료하기.
        printf("프로그램을 종료합니다.");
        return 0;
    }
}

```

```

    idxCnt++;
    *(numList + idxCnt) = nextInput;

    // 입력 받은 숫자로 배열 전체 지정 및 출력하기.
    for (i = 0; i < 3; i++) {
        for (j = 0; j < 4; j++) {
            numAry[i][j] = *(numList + idxCirculation);
            if (*(numList + idxCirculation) == 0) {
                idxCirculation = 0;
                numAry[i][j] = *(numList + idxCirculation);
                idxCirculation++;
            }else{
                idxCirculation++;
            }
            printf("%d ", numAry[i][j]);
        }
        printf("\n");
    }
    idxCirculation = 0;
    printf("\n\n");

}

free(numList);

return 0;
}

```

[실행결과]

Test Case #1

```
C:\Users\WordkW\Github\WROE x + v
input : 3
3 3 3 3
3 3 3 3
3 3 3 3

next input : 5
3 5 3 5
3 5 3 5
3 5 3 5

next input : 1
3 5 1 3
5 1 3 5
1 3 5 1

next input : |
```

Test Case #2

```
C:\Users\WordkW\Github\WROE x + v - □ x
input : 5
5 5 5 5
5 5 5 5
5 5 5 5

next input : 3
5 3 5 3
5 3 5 3
5 3 5 3

next input : 2
5 3 2 5
3 2 5 3
2 5 3 2

next input : 1
5 3 2 1
5 3 2 1
5 3 2 1

next input : 4
5 3 2 1
4 5 3 2
1 4 5 3

next input : |
```

Test Case #3

```
C:\Users\Wordk\Github\WROE x + v
input : 11
11 11 11 11
11 11 11 11
11 11 11 11

next input : 4
11 4 11 4
11 4 11 4
11 4 11 4

next input : 9
11 4 9 11
4 9 11 4
9 11 4 9

next input : 8
11 4 9 8
11 4 9 8
11 4 9 8

next input : 2
11 4 9 8
2 11 4 9
8 2 11 4

next input : |
```

3. HW_003

2024년 7월 8일 배정받은 C언어 6일차 과제 3번은 8시간 넘게 스스로 해결하기 위해 시도해 보았으나, 끝내 수행하지 못했습니다. C언어 6일차 교육을 담당해 주신 조동희 선배님께 소장하고 있는 C언어 자료구조 책에 설명된 선형 리스트, 연결 리스트 개념을 공부하고 이해한 내용을 토대로 과제를 수행해도 되는지 여쭙보았습니다. 코드를 그대로 작성하는 것이 아닌, 공부하고 이해하여 작성하는 모든 것은 괜찮다는 답변을 받을 수 있었습니다. 리스트 내용을 이해하려고도 시간을 투자하였으나, 개념을 이해하는 데 있어서 많은 어려움이 있습니다. 특히, 구조체 내부에 포인터 형태의 구조체가 멤버 변수로 있는데, 이 멤버 변수 내부에 있는 또 다른 멤버 변수를 어떻게 참조해야 하는지 모르겠습니다. 관련 내용을 찾기 위해 여러 방안을 사용해 보았으나, 끝내 방법을 찾지 못했습니다. C언어 과제 3번은 C언어에서 연결 리스트를 구현하는 알고리즘에 대하여 로빗 활동 외에 개별적으로 공부하도록 하겠습니다. 공부한 내용은 가능한 Obsidian에 기록하여 제대로 이해할 수 있도록 실습을 통해 공부해 이해하겠습니다.

현재까지 제가 디자인한 과제 3의 알고리즘 로직입니다.

노드1이 선언됩니다. 노드1의 data값은 NULL로 처리됩니다. (구현 완료)

Insert 명령어를 사용자로 부터 입력 받을 시, 노드2가 선언됩니다. (구현 실패 ~> 참조 관련 휴먼 에러로 추정 중)

노드2의 data값은 NULL로 처리됩니다.

해당 기능들은 현재 주석 처리 하였습니다.

현재 구현된 알고리즘은 다음과 같습니다.

Insert, insert_first, delete 등의 명령어(cmd)가 입력되면 각 조건에 따라 명령어가 정상적으로 분류 되는지 나타냅니다.

[소스코드]

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

typedef struct _Node {
    int data;
    struct _Node* next; // 다음 노드를 가리킴.
}Node;

typedef struct _LinkedList {
    Node* head;
    Node* tail;
    Node* Cur;
    int size;
}LinkedList;

void Inserting(LinkedList* list);

int main() {
    // 단순 연결 리스트 구조체 선언하기.
    LinkedList* linkedList = (LinkedList*)malloc(sizeof(linkedList) * 10);
    // 노드1 생성하기.
    Node* node = (Node*)malloc(sizeof(node));

    // 변수 선언하기.
    char* inputCmd = (char*)malloc(sizeof(char) * 20);

    // 노드1의 현재 값은 NULL 상태임.
    /*node->data = NULL;
    printf("size : %d\n\n", node->data);*/

    // 노드1 연결하기
    /*printf("Bit structure Data : %d\n\n", node->data);*/

    // 명령어(cmd)을 입력 받고, 각 명령에 따라 정해진 작업을 진행함.
```



```

while (1){
    scanf("%s", inputCmd);

    if (strcmp(inputCmd, "insert") == 0) {
        // 두 번째 노드(노드2)를 기존 노드(노드1)의 꼬리 뒤에 오게끔 함.
        Node* node = linkedList->tail;
        // 노드2의 값을 NULL로 지정함. (현재 상황 : node1->node2)
        node->next->data = NULL;
        printf("%d %d", node->data, node->next->data);
        printf("insert!");
    }else if (strcmp(inputCmd, "insert_back") == 0) {
        printf("insert_back!");
    }else if (strcmp(inputCmd, "insert_first") == 0) {
        printf("insert_first!");
    }else if (strcmp(inputCmd, "delete") == 0) {
        printf("delete!");
    }else if (strcmp(inputCmd, "delete_first") == 0) {
        printf("delete_first!");
    }else if (strcmp(inputCmd, "delete_back") == 0) {
        printf("delete_back!");
    }else if (strcmp(inputCmd, "get_entry") == 0) {
        printf("get_entry!");
    }else if (strcmp(inputCmd, "get_length") == 0) {
        printf("get_length!");
    }else if (strcmp(inputCmd, "print_list") == 0) {
        printf("print_list!");
    }else if (strcmp(inputCmd, "reverse") == 0) {
        printf("reverse!");
    }else if (strcmp(inputCmd, "Data") == 0) {
        printf("Data!");
    }

    printf("\n");
}

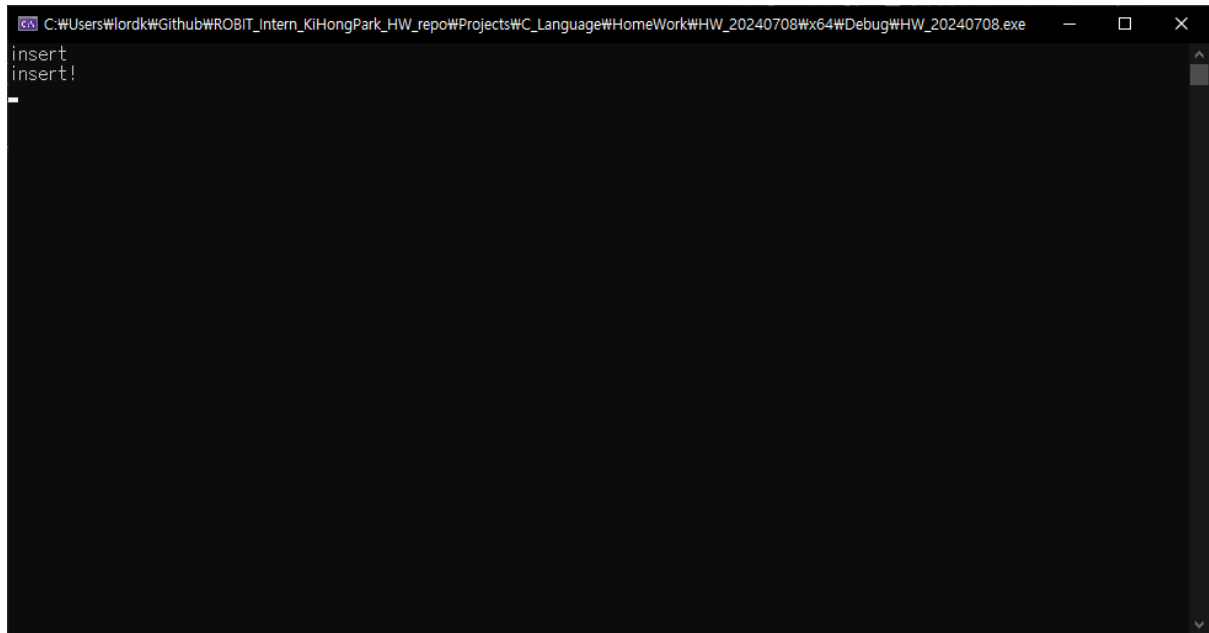
return 0;
}

// 1. insert 함수
void Inserting(LinkedList* list) {
}

```

[실행결과]

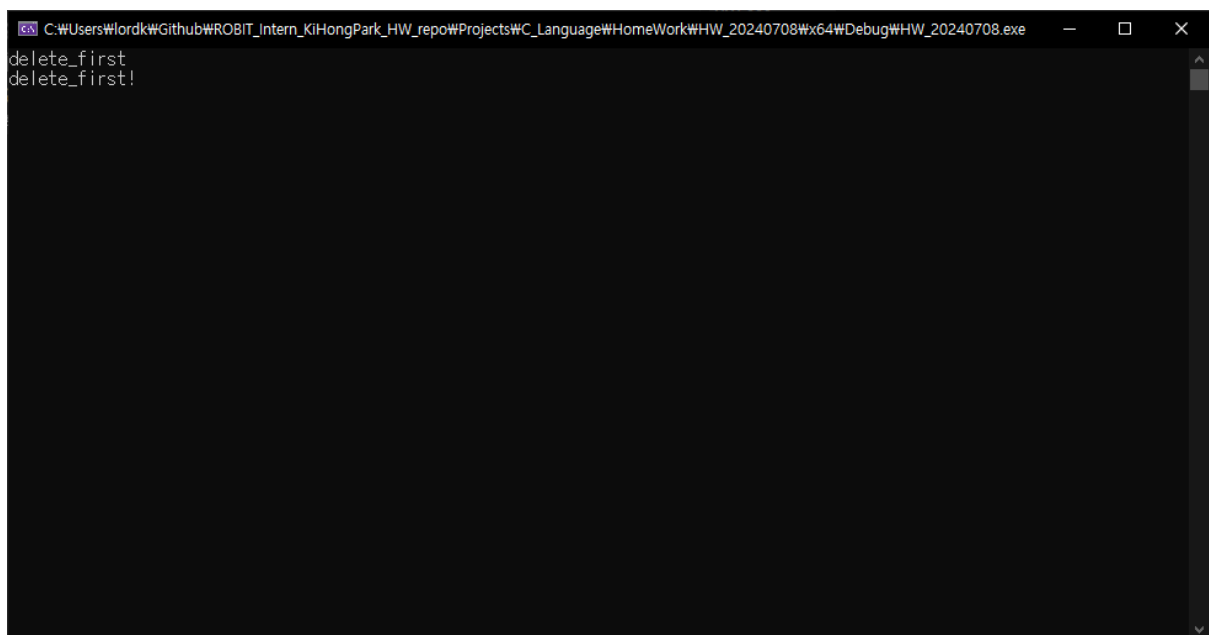
Test Case #1



```
C:\Users\lordk\Github\ROBIT_Intern_KiHongPark_HW_repo\Projects\C_Language\HomeWork\HW_20240708\x64\Debug\HW_20240708.exe
insert
insert!

```

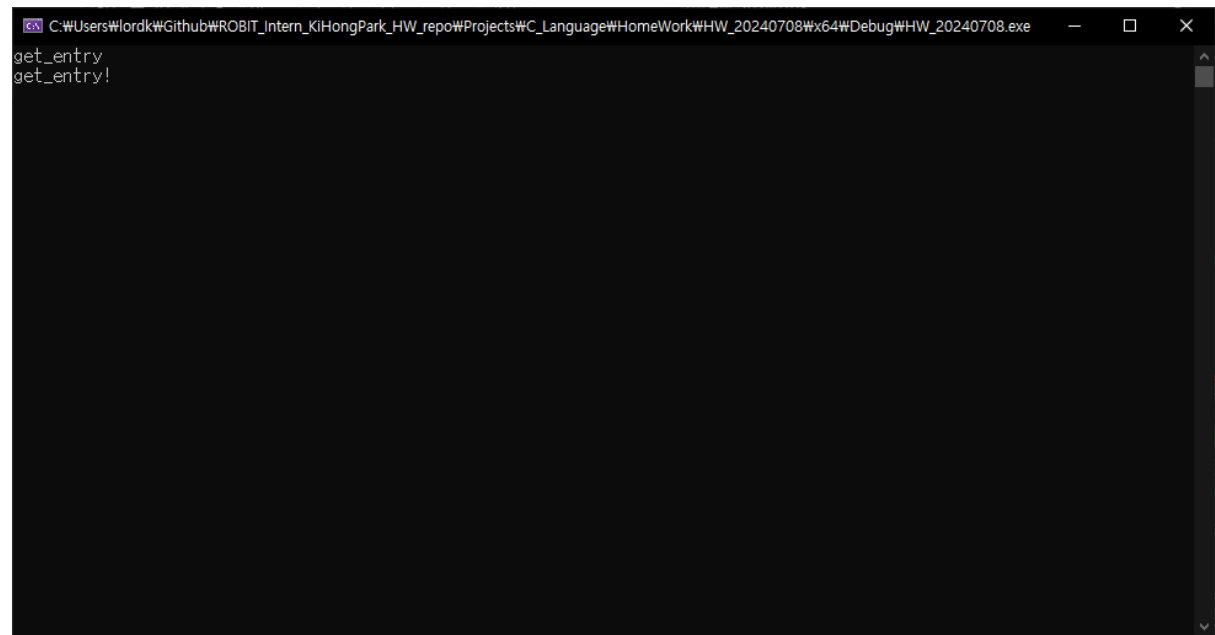
Test Case #2



```
C:\Users\lordk\Github\ROBIT_Intern_KiHongPark_HW_repo\Projects\C_Language\HomeWork\HW_20240708\x64\Debug\HW_20240708.exe
delete_first
delete_first!

```

Test Case #3



A screenshot of a Windows command prompt window. The title bar at the top shows the file path: `C:\Users\lordk\Github\ROBIT_Intern_KiHongPark_HW_repo\Projects\C_Language\HomeWork\HW_20240708\x64\Debug\HW_20240708.exe`. The window has standard Windows window controls (minimize, maximize, close) on the right. The command prompt area is black with white text. The first line shows the command `get_entry` being entered. The second line shows the output `get_entry!`. A vertical scrollbar is visible on the right side of the command prompt area.

```
C:\Users\lordk\Github\ROBIT_Intern_KiHongPark_HW_repo\Projects\C_Language\HomeWork\HW_20240708\x64\Debug\HW_20240708.exe
get_entry
get_entry!
```