PART 13 동적할당 연습문제 풀이

1. 다음 중 정적 변수가 아닌 것은? ③

   ① 전역 변수

   ② 파일범위 변수

   ③ 매개변수

   ④ `static` 지역 변수

2. 다음 중 자유 저장공간에 할당되는 변수는? ④

   ① 지역변수

   ② 매개변수

   ③ 파일범위 변수

   ④ 동적 할당 변수

3. 다음 메모리 관리 함수 중에서 다른 것들과 성격이 다른 하나는? ③

   ① `calloc`

   ② `malloc`

   ③ `free`

   ④ `realloc`

4. 메모리 관리 함수에 대한 설명으로 옳지 <u>못한</u> 것은? ①

   ① `malloc`은 원소 하나를 할당하는 데 사용할 수 있지만 배열 할당에는 사용할 수 없다.

   ② `malloc`은 메모리 할당은 수행하지만 할당된 메모리 초기화는 수행하지 않는다.

   ③ `calloc`은 같은 크기의 메모리 블록을 여러 개 할당하는 데 사용된다.

   ④ `realloc`은 이미 할당되어 있던 메모리의 크기를 확장하는 데 사용된다.

5. 다음 중 `assert`에 대한 설명으로 옳지 <u>못한</u> 것은? ③

   ① `assert`는 수행 중 만족해야 할 조건을 기술할 때 사용한다.

   ② `assert`를 사용하려면 `<assert.h>`를 `#include`해야 한다.

   ③ `assert`를 끄려면 전처리기 상수 `NODEBUG`를 정의하면 된다.

   ④ `assert`는 실행 중에 검사하므로 실행 시 성능 저하의 원인이 될 수 있다.

6. 다음 중 스택(stack)의 구조로 적합한 것은? ④

   ① FIFO(first in, first out)

② FEFO(first expiry, first out)

③ GIGO(garbage in, garbage out)

④ LIFO(last in, first out)

7. 다음 중 자기 참조 구조체를 사용하는 이유로 가장 적합한 것은? ③

① 피드백 회로를 구성하기 위해서

② 자유 저장공간을 활용하기 위해서

③ 저장해야 하는 데이터 개수를 미리 알 수 없기 때문에

④ 수행 중 자료형 검사를 통해 프로그램 안전성을 높이기 위해서

8. 메모리를 할당하는 표준 라이브러리 함수 `malloc`과 할당된 메모리를 해제하는 표준 라이브러리 함수 `free`의 프로토타입을 기술하라.

```
void *malloc(size_t size);
void free(void *ptr);
```

9. 앞서 설명한 것처럼 `calloc` 함수의 프로토타입은 다음과 같다.

```
void* calloc(size_t nobj, size_t size);
```

`calloc` 함수를 이용하여 앞서 예시한 프로그램 `prNint.c`를 다시 작성하라.

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

int main()
{
    int *p;     // 동적 할당된 블록을 가리킬 포인터
    int n, i;

    printf("몇 개의 정수를 입력하고 싶으십니까? ");
    scanf("%d", &n);

    p = (int *) calloc(n, sizeof(int));

    printf("%d 개의 정수를 입력해 주세요. ", n);
    for (i = 0; i < n; ++i)
        scanf("%d", &p[i]);
    printf("입력한 정수를 역순으로 출력합니다. ");
```

```
    for (i = n-1; i >= 0; --i)
        printf("%d ", p[i]);
    printf("\n");

    return 0;
}
```

10. 동적으로 할당할 수 있는 공간(자유저장 공간)의 최대 크기를 측정하는 프로그램을 작성한다.

① 반복문 내에서 malloc을 호출하여 1KB 단위로 메모리를 할당한다. malloc의 결과 값을 검사하여 메모리 할당에 실패할 때까지 malloc 호출을 반복한다. malloc을 호출할 때마다 malloc 호출 회수를 센다. 메모리 할당에 실패하면 malloc 호출 회수를 이용하여 자유저장 공간의 크기를 개략적으로 산출한다.

다음 코드를 수행하면 된다. 가용 메모리 부족으로 인해 시스템이 불안해질 수 있으므로 다른 프로그램은 모두 닫고 수행하는 것이 좋다.

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

int main()
{
    void *p;
    int n = 0;

    while ((p = malloc(1024)) != NULL)
        n++;

    printf("malloc(1024) 호출 횟수: %d\n", n);

    return 0;
}
```

② 할당 단위를 4 바이트로 변경하여 같은 실험을 반복하라. 1KB 단위로 실험한 결과와 비교하여 자유저장 공간의 크기 추정치를 비교하라. 이런 결과가 나온 이유를 설명하라.

위 코드에서 밑줄 그은 1024를 4로 바꾸어 측정한다. 메모리가 크면 매우 오래 걸

리므로 주의한다.

11. 문자열을 n개 입력받아서 이들 문자열을 사전 순으로 정렬하는 프로그램을 작성하라. 이 프로그램은 처음에 문자열 개수 n을 입력받는다. 그 다음 각 문자열을 저장하는데 필요한 공간을 할당한다. 문자열의 최대 길이는 #define 상수 MAX_LENGTH로 정한다. 문자열을 읽어 들일 때에는 MAX_LENGTH 길이의 char형 배열에 읽어 들이지만, 일단 읽은 후에는 각 문자열 저장에 필요한 만큼만 메모리를 할당하여 문자열을 저장한다. 사전순으로 정렬하기 위해서 라이브러리 함수 qsort를 이용한다.

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

#define MAX_LENGTH  256

void prstrs(char *ss[], int n);
int cmp(const void *, const void *);

int main()
{
    char **strs;
    int n = 0, i = 0;
    char str[MAX_LENGTH];

    printf("입력할 문자열은 몇 개인가요? ");
    scanf("%d", &n);
    fgets(str, MAX_LENGTH, stdin);      // 남은 행 읽기

    assert(n > 0);
    strs = (char **) calloc(n, sizeof(char *));

    while (fgets(str, MAX_LENGTH, stdin) != NULL) {
        int len = strlen(str);
        str[len-1] = '\0';
        strs[i] = (char *) calloc(strlen(str) + 1, sizeof(char));
        strcpy(strs[i], str);
        i++;
        if (i >= n) break;
    }
```

```
    qsort(strs, i, sizeof *strs, cmp);
    prstrs(strs, i);

    return 0;
}

void prstrs(char *ss[], int n)
{
    for (int i = 0; i < n; i++)
        puts(ss[i]);
}

int cmp(const void *p, const void *q)
{
    char *s = *(char **)p, *t = *(char **)q;
    return strcmp(s, t);
}
```

12. 연결 리스트를 이용하여 출석부를 관리하는 프로그램을 작성한다. 출석부는 학생 이름의 가나다 순서로 정렬 유지되어야 하며 새로운 학생의 학번과 이름이 입력될 때마다 정렬을 유지하도록 학생 정보를 적합한 위치에 삽입해야 한다. 프로그램은 학생 추가, 검색, 삭제를 위한 메뉴를 제공해야 한다.

```
#include <stdio.h>
#include <assert.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>

#define NAME_LEN  256

typedef struct _st_node {
    char *name;
    int id;
    struct _st_node *next;
} Student;

typedef enum menu {ADD=1, DEL, SID, SNM, EXIT} Menu;

int int_or_retry();
```

```
Menu select_menu()
{
    int menu = 0;
    printf("----- 출석부 관리 시스템 -----\n");
    printf("1) 학생 정보 추가\n");
    printf("2) 학생 정보 삭제\n");
    printf("3) 학생 정보 검색(학번으로 검색)\n");
    printf("4) 학생 정보 검색(이름으로 검색)\n");
    printf("5) 종료\n");
    printf("메뉴 선택 ---->  ");
    menu = int_or_retry();
    return menu;
}

int int_or_retry()
{
    char input[NAME_LEN];
    scanf("%s", input);
    if (isdigit(input[0]))
        return atoi(input);
    else
        return int_or_retry();
}

void add_student(Student **slist);
void del_student(Student **slist);
void find_student_by_id(Student *slist);
void find_student_by_name(Student *slist);
void print_list(Student *slist);

int main()
{
    Student *slist = NULL;
    Menu menu;

    do {
        menu = select_menu();
        switch (menu) {
            case ADD:
                add_student(&slist);
```

```
                print_list(slist);
                break;
            case DEL:
                del_student(&slist);
                print_list(slist);
                break;
            case SID:
                find_student_by_id(slist);
                break;
            case SNM:
                find_student_by_name(slist);
                break;
            default:
                printf("잘못된 선택입니다.\n:");
            case EXIT:
                break;
        }
    } while (menu != EXIT);

    return 0;
}

Student *mk_student(char *name, int id)
{
    Student *record = (Student *)malloc(sizeof(Student));
    record->id = id;
    record->name = (char *)malloc(strlen(name)+1);
    strcpy(record->name, name);
    record->next = NULL;
    return record;
}

void swap_record(Student *p, Student *q)
{
    int id = p->id;
    char *name = p->name;
    p->id = q->id;
    p->name = q->name;
    q->id = id;
    q->name = name;
}
```

```
void add_student(Student **slist)
{
    int id;
    char name[NAME_LEN];
    printf("추가할 학생의 학번과 이름을 입력하세요. ");
    scanf("%d", &id);
    scanf("%s", name);
    if (*slist == NULL) {   // Entering the first record
        *slist = mk_student(name, id);
        return;
    }
    Student *p = *slist, *q = p->next;
    for (; p != NULL; p = q, q = q->next) {
        if (strcmp(name, p->name) < 0) {    // Enter before p
            Student *n = mk_student(name, id);
            swap_record(p, n);
            p->next = n;
            n->next = q;
            return;
        }
        if (q == NULL) {
            p->next = mk_student(name, id);
            return;
        }
    }
}

void del_student(Student **slist)
{
    int id;
    printf("삭제할 학생의 학번을 입력하세요. ");
    scanf("%d", &id);
    Student *p = *slist;
    if (p == NULL) {    // No record
        printf("삭제할 학생이 없습니다.\n");
        return;
    }
    if (p->id == id) {  // Deleting the first record
        printf("[%d %s] 정보를 삭제합니다.\n", p->id, p->name);
        *slist = p->next;
```

```
        free(p->name);
        free(p);
        return;
    }
    Student *q = p->next;
    for (; q != NULL; p = q, q = q->next) {
        if (q->id == id) {  // Found the record
            printf("[%d %s] 정보를 삭제합니다.\n", q->id, q->name);
            p->next = q->next;
            free(q->name);
            free(q);
            return;
        }
    }
    printf("정보가 일치하는 학생이 없습니다.\n");
}

void find_student_by_id(Student *slist)
{
    int id;
    printf("검색할 학생의 학번을 입력하세요. ");
    scanf("%d", &id);
    for (Student *p = slist; p != NULL; p = p->next) {
        if (p->id == id) {
            printf("[%d %s] 학생을 찾았습니다.\n", p->id, p->name);
            return;
        }
    }
    printf("학번 %d와(과) 일치하는 학생이 없습니다.\n", id);
}

void find_student_by_name(Student *slist)
{
    char name[NAME_LEN];
    printf("검색할 학생의 이름을 입력하세요. ");
    scanf("%s", name);
    for (Student *p = slist; p != NULL; p = p->next) {
        if (strcmp(p->name, name) == 0) {
            printf("[%d %s] 학생을 찾았습니다.\n", p->id, p->name);
        }
    }
```

```
    printf("더 이상 이름이 일치하는 학생이 없습니다.\n");
}

void print_list(Student *slist)
{
    if (slist) {
        printf("    %8d %s\n", slist->id, slist->name);
        print_list(slist->next);
    }
}
```