5장. 응용예제 답

응용예제 05

핀볼 구슬 게임의 점수는?

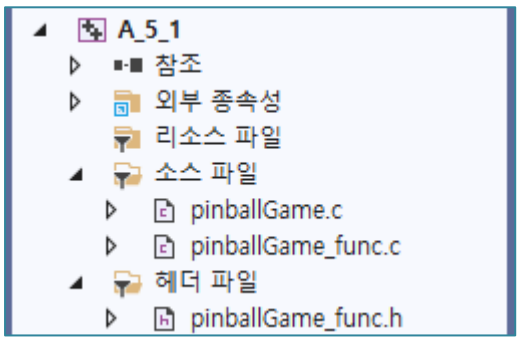
④ 입출력 예시 **수정**

입력예시:

```
0 0 0 0 0 0 0 2 0 4 2 0 5 3 1
30 20 40 10 20
5 1, 3 5, 3 2, 2 3, 4 2, 3 4, 5 4, 5 3, 2 1, 1 3
```

출력예시:

```
( 270점   140점   40점   50점   160점 )
```



pinballGame_func.h

```
#pragma once
#define BOARD_ROW 10
#define BOARD_COL 5    //핀볼게임보드의 열 수. 스택의 갯수
#define MAX_NEW_BALL 10 //핀볼 게임 시작 후, 입력된 새 공 갯수

typedef struct stackNode { // 스택의 노드를 구조체로 정의
        int data;
        struct stackNode* link;
} stackNode;


void push_pinBall(int index, int ball);
int pop_pinBall(int index);
int isEmpty_pinBall_STACK(int index);
int peek_pinBall(int index);
int sum_pinBall(int index);
```

pinballGame_func.c

```c
#include <malloc.h>
#include "pinballGame_func.h"
// 예제 5-2의 연결스택을 변형하여 사용

stackNode* pinBall_STACK[BOARD_COL] = { 0 };

// 핀볼 보드 스택에 공 삽입하기
void push_pinBall(int index, int ball) {
	stackNode* temp = (stackNode*)malloc(sizeof(stackNode));
	temp->data = ball;
	temp->link = pinBall_STACK[index]; // 삽입 노드를 board_top[index]의 위에 연결
	pinBall_STACK[index] = temp;		// board_top[index] 위치를 삽입 노드로 이동
}

// 핀볼 보드 스택에서 top에 있는 공 삭제
int pop_pinBall(int index) {
	int ball;
	stackNode* temp = pinBall_STACK[index];

	if (isEmpty_pinBall_STACK(index))  // 스택이 공백 리스트인 경우
		return 0;
	else {                                                          // 스택이
공백 리스트가 아닌 경우
		ball = temp->data;
		pinBall_STACK[index] = temp->link;
		free(temp);                                                 // 삭제된
노드의 메모리 반환
		return ball;                                            // 삭제된 원소 반환
	}
}

// 핀볼 보드 스택의 top에 있는 공 검색
int peek_pinBall(int index) {
	if (!isEmpty_pinBall_STACK(index))		//스택이 공백 리스트가 아닌 경우
		return pinBall_STACK[index]->data; // 현재 top의 원소 반환
	return 0;
}

// 핀볼 보드 스택이 공백 상태인지 확인하는 연산
int isEmpty_pinBall_STACK(int index) {
	if (pinBall_STACK[index] == NULL)
		return 1;
	return 0;
}

//index에 해당하는 핀볼 보드 열에 있는 공 번호의 합 구하기
int sum_pinBall(int index) {
	int sum = 0;
	stackNode* temp = pinBall_STACK[index];

	if (temp == NULL)
		return 0;
	else {
```

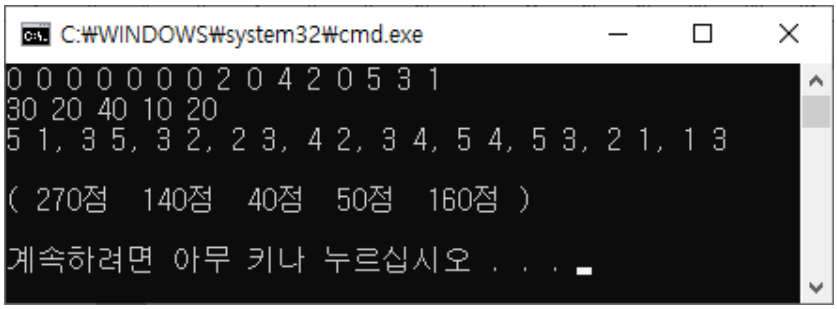
```
                do {
                        sum += temp->data;
                        temp = temp->link;
                } while (temp != NULL);
                return sum;
        }
}
```

pinballGame.c

```
//응용예제05. 핀볼게임 점수는?
#define _CRT_SECURE_NO_WARNINGS
#pragma warning(disable : 6031) //scanf의 반환값없음warning C6031을 해제하기 위해 추가.
#include<stdio.h>
#include "pinballGame_func.h"

int main(void)
{
        int pinBall[BOARD_ROW][BOARD_COL] = { 0 };
        int scores[BOARD_COL] = { 0 },  playBalls[MAX_NEW_BALL][2] = { 0 };
        int ball_num = 0, board_num = 0, i, j ;

        // 1) 핀볼의 시작상태 세줄 입력받기
        for (i = 7; i < BOARD_ROW; i++)
                for (j = 0; j < BOARD_COL; j++)
                        scanf("%d", &pinBall[i][j]);

        // 2) 핀볼 세로칸 점수 입력받기
        for (i = 0; i < BOARD_COL; i++)
                scanf("%d", &scores[i]);

   // 3) 핀볼게임 새 구슬(구슬번호 playBalls[i][0], 보드의 세로칸 번호playBalls[i][1]) 정보 10개
입력 받기
        for (i = 0; i < MAX_NEW_BALL; i++)
                scanf("%d %d,", &playBalls[i][0], &playBalls[i][1]);

        // 4) 핀볼 보드를 스택으로 구성하기
        for (i = 0; i < BOARD_COL; i++)
                for (j = BOARD_ROW - 1; j >= 0; j--) { //pinBall 배열의 i번 열을

                        if (pinBall[j][i] != 0)
                                push_pinBall(i, pinBall[j][i]);   // i번 스택으로 구성
                }

        // 5) 핀볼 게임 하기
        for (i = 0; i < MAX_NEW_BALL; i++) {
                if (playBalls[i][0] == peek_pinBall(playBalls[i][1]-1)) //구슬을 넣을 열의 top
원소와 새구슬의 번호가 같으면, 스택 원소 삭제
                        pop_pinBall(playBalls[i][1] -1); //입력받은 열 번호(playBalls[i][1])인
1~5를 배열 인덱스 0~4로 변경
                else     //그렇지않으면, 스택에 구슬 삽입
                        push_pinBall(playBalls[i][1]-1, playBalls[i][0]);
```

```
        }

        // 6) 점수 계산하기
        printf("₩n(");
        for (i = 0; i < BOARD_COL; i++) {
                        printf(" %d점 ", scores[i] * sum_pinBall(i));
        }
        printf(")₩n₩n");

        return 0;
}
```

[실행 화면]

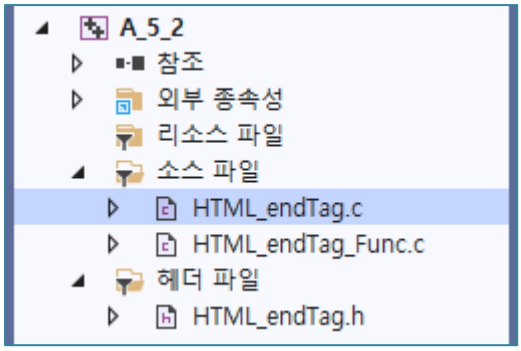


응용예제
06

종료 태그를 자동 입력하는 HTML 편집기

예제 5-2를 수정하여 작성.

스택 노드의 data 필드에는 토큰 문자열에 대한 포인터를 저장.



HTML_endTag.h

```
#pragma once
typedef char* element;            // 스택 원소(element)의 자료형을 char*로 수정 !! 토큰 문자열에 대한 포인터 저장

typedef struct stackNode {        // 스택의 노드를 구조체로 정의
        element data;
        struct stackNode* link;
} stackNode;

stackNode* top;

int isStackEmpty();
void push(element item);
element pop();
void printStack();
```

HTML_endTag.c

```c
#include<stdio.h>
#include "HTML_endTag.h"
// 예제 5-2 ////////////////////////////////////////////////////////////////////

// 스택이 공백 상태인지 확인하는 연산
int isStackEmpty() {
        if (top == NULL) return 1;
        else return 0;
}

// 스택의 top에 원소를 삽입하는 연산
void push(element item) {
        stackNode* temp = (stackNode*)malloc(sizeof(stackNode));

        temp->data = item;
        temp->link = top;     // 삽입 노드를 top의 위에 연결
        top = temp;           // top 위치를 삽입 노드로 이동
}

// 스택의 top에서 원소를 삭제하는 연산
element pop() {
        element item;
        stackNode* temp = top;

        if (isStackEmpty()) {            // 스택이 공백 리스트인 경우
                printf("₩n₩n Stack is empty !₩n");
                return 0;
        }
        else {                                  // 스택이 공백 리스트가 아닌 경우
                item = temp->data;
                top = temp->link;        // top 위치를 삭제 노드 아래로 이동
                free(temp);                     // 삭제된 노드의 메모리 반환
                return item;             // 삭제된 원소 반환
        }
}

// 스택의 원소를 top에서 bottom 순서로 출력하는 연산
void printStack() {
        stackNode* p = top;
        printf("₩n STACK [ ");
        while (p) {
                printf("%s ", p->data);
                p = p->link;
        }
        printf("] ");
}
//////////////////////////////////////////////////////////////////// 예제 5-2 //
```

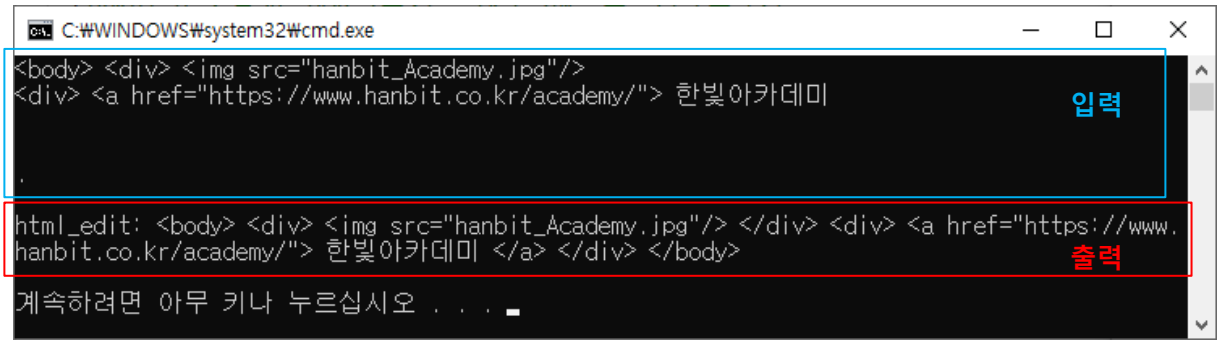
HTML_endTag.c

```
//응용예제06. 종료 태그를 자동 입력하는 HTML 편집기
#define _CRT_SECURE_NO_WARNINGS  //strtok, scanf 보안 경고 해제
#pragma warning(disable : 6031) //scanf의 반환값없음warning C6031을 해제하기 위해 추가.
#include<stdio.h>
#include<malloc.h>
#include <string.h>
#include "HTML_endTag.h"
#define MAX 100

int main(void) {
	char html[MAX] = { 0 };
	char* p, * q, * token, * stack_token, html_edit[MAX * 10] = { '₩0' }, * r, * str;
	int i = 0;
	r = html_edit;  //출력할 html  저장용

	while (1) {
		// html 입력
		//printf("html 입력 >> ");  //작업 단계 확인용! ////////////////////////////
		gets(html);

		// 첫번째 토큰 만들기
		token = strtok(html, " ");

		while (token != NULL) {
			// 토큰을 출력 문자열에 복사
			if (token[0] != '.') {
				strcpy(r, token);  //포인터 r 위치에 토큰 복사하기
				r += strlen(token); //복사한 토큰 끝으로 포인터 r 이동
				*r = ' '; r++; //띄어쓰기 추가하고, 포인터 r을 다음 칸으로 위치
조정
				//printf("₩n작업중_html_edit: %s₩n₩n", html_edit);  //현재까지
작업한 출력 문자열 출력 : 확인용!//////////////////////////
			}

			p = token;
			q = p + strlen(token);  //토큰 끝(NULL)에 포인터 q 설정

			if (*p == '<') {  // '<'로 시작하는 토큰인 경우,

				if (*(q - 2) == '/') {//토큰의 끝에서 두번째 위치에 있는 문자
확인 => 종료태그 포함 태그인가?
					// 작업 없이 블록을 나가서, 새 토큰 읽기 수행
				}
		// => 종료태그 포함 태그가 아닌 경우 : 종료태그를 만든 stack_token을 스택에 push
				else {
					if (*(q - 1) == '>') { //시작태그이고, 속성이 연결되지
않은 경우
```

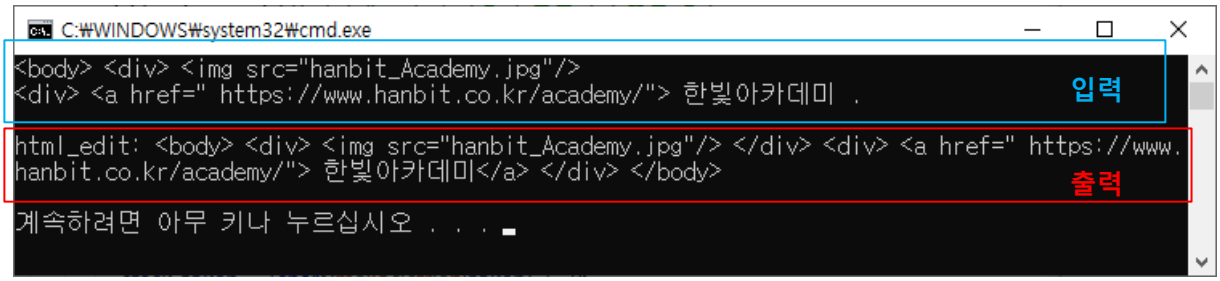
```
                                        stack_token = (char*)malloc(strlen(token) +
1);
                                        stack_token[0] = '<';
                                        stack_token[1] = '/';
                                        for (i = 2, p += (i - 1); p <= q; i++, p++) {
                                                stack_token[i] = *p;
                                        }
                                }
                                else {  //시작태그이고, 속성이 연결되지 있는 경우 =>
토큰 끝에 '>' 추가하여 스택 토큰 생성
                                        stack_token = (char*)malloc(strlen(token) +
2);
                                        stack_token[0] = '<';
                                        stack_token[1] = '/';
                                        for (i = 2, p += (i - 1); p < q; i++, p++) {
                                                stack_token[i] = *p;
                                        }
                                        stack_token[i] = '>';
                                        stack_token[i + 1] = '₩0';
                                }
                                // 생성한 종료태그를 스택에 삽입
                                push(stack_token);
                                //printStack(); //현재 스택 내용 출력 : 확인용!////////
                        }
                } //if (*p == '<') {  // '<'로 시작하는 토큰인 경우,
                else {  //  '<'로 시작하지 않는 경우 : 태그 안에 있는 속성 토큰인 경우
                        if (*(q - 2) == '/') {     //토큰의 끝에서 두번째 위치에 있는
문자 확인 => 종료태그 포함 태그인가?
                                pop(); //종료태그가 포함되어있으므로, 스택에 만들어둔
종료태그는 필요없으므로 삭제
                        }
                        else {
                                if (token[0] == '.') { //토큰이 마침표인 경우
                                        r -= 1;  //마지막 복사한 '.' 토큰 삭제
                                // 스택에 남아있는 종료태그를 모두 꺼내어 출력문자열에 복사
                                        while (!isStackEmpty()) {
                                                str = pop();
                                                // 토큰을 출력 문자열에 복사
                                                strcpy(r, str); //포인터 r 위치에
토큰 복사하기
                                                r += strlen(str);  //복사한 토큰
끝으로 포인터 r 이동
                                                *r = ' '; r++;     //띄어쓰기
추가하고, 포인터 r을 다음 칸으로 위치 조정
                                        }

                                        // 최종 완성된 출력 문자열 출력하기
                                        printf("₩nhtml_edit: %s₩n₩n", html_edit);

                                        return 0;  // 프로그램 종료
                                }
```

```
                                    }
                            } //else { //'<'로 시작하지 않는 경우 : 태그 안에 있는 속성 토큰인 경우

                    //if~else 처리 후, 공통 처리할 것. -> 새토큰 읽기
                            token = strtok(q + 1, " ");      // 다음 토큰을 잘라서 포인터를 반환

                    } //while (token != NULL)

                    //token 이 NULL인 경우 => 종료태그 삽입할 자리
                    str = pop();
                    // 토큰을 출력 문자열에 복사
                    strcpy(r, str); //포인터 r 위치에 토큰 복사하기
                    r += strlen(str);  //복사한 토큰 끝으로 포인터 r 이동
                    *r = ' '; r++;  //띄어쓰기 추가하고, 포인터 r을 다음 칸으로 위치 조정
                    //printf("₩n작업중_html_edit: %s₩n₩n", html_edit);  //현재까지 작업한 출력 문자열
출력 : 확인용! ///////////
            } //while (1)
}
```

[입출력 예시 1] 실행결과 >



[입출력 예시 2] 실행결과 >



[HTML_endTag.c]에서 작업 확인용 “printf()” 주석을 해제한 후, [입출력 예시 1] 실행결과 >