자료구조 실습과제 10
- 연결리스트를 이용한 스택, 큐 구현

- 솔루션 및 프로젝트 명칭 : Proj_10_이름이니셜

- 제출방법 : 아래 문제를 해결하기 위한 프로그램을 구현한 후 컴파일 및 실행한 후, 오류가 없으면 메뉴에서 **솔루션 정리를 수행한 후 윈도우 탐색기에서 솔루션 폴더를 찾아 압축하여** E-class에 올림

각 문제를 테스트하기 위하여 아래와 같이 구현하시오.

```
#define _CRT_SECURE_NO_WARNINGS

#define PROB 1 // 각각의 문제를 구현하고 해당 문제 번호를 변경하여 테스트

#if PROB == 1
// 1번 문제를 해결하기 위한 프로그램 작성

#elif PROB == 2
// 2번 문제를 해결하기 위한 프로그램 작성

#endif
```

문제 1) 연결리스트를 이용한 스택 구현(교재 7.5절 참고)

연결리스트를 이용하여 스택을 구현하시오. 배열을 이용하여 구현한 예에서와 같이 스택 운영을 위한 기본연산인 is_empty(), is_full(), push(), pop(), peek(), print_stack() 연산을 구현하시오.
스택에 저장되는 데이터는 아래에 정의된 3차원 세계에서의 좌표이다.

```
typedef struct WORLD_COORDINATE {
        int x;
        int y;
        int z;
} element;

typedef struct STACKNODE {
        element data;
        struct STACKNODE* link;
} StackNode;
```

```
typedef struct {
        StackNode* top;
} LinkedlistStack;
```

is_full()은 항상 0을 반환한다. push() 함수에서 스택 노드의 메모리 할당이 안될 경우 에러 처리 한다.
print_stack() 함수를 수행하면 아래 실행 예와 같이
=======스택내용=======
.....
======================
형식으로 스택에 저장된 내용과 Top 포인터의 위치를 표현한다.
스택이 빈 상태에서 pop을 할 경우 강제 종료를 하지 않고 정수 중 가장 작은 값인 {INT_MIN, INT_MIN, INT_MIN} 값을 반환한다.
나머지 연산은 스택의 추상 데이터 형식에 저장된 함수의 기능과 동일하다.

실행 예

```
int main(void)
{

        ...

        init(&s);
        print_stack(&s);
        item.x = 1; item.y = 1; item.z = 1;
        printf("Push item\n");
        push(&s, item); print_stack(&s);

        item.x = 2; item.y = 2; item.z = 2;
        printf("Push item\n");
        push(&s, item); print_stack(&s);

        item.x = 3; item.y = 3; item.z = 3;
        printf("Push item\n");
        push(&s, item); print_stack(&s);

        item.x = 4; item.y = 4; item.z = 4;
        printf("Push item\n");
        push(&s, item); print_stack(&s);

        printf("PoP item\n");
        item = pop(&s);
        printf("PoP된 data : (%d, %d, %d) \n", item.x, item.y, item.z);
        print_stack(&s);

        printf("PoP item\n");
        item = pop(&s);
        printf("PoP된 data : (%d, %d, %d) \n", item.x, item.y, item.z);
        print_stack(&s);

        printf("PoP item\n");
        item = pop(&s);
        printf("PoP된 data : (%d, %d, %d) \n", item.x, item.y, item.z);
        print_stack(&s);

        printf("PoP item\n");
```

```
        item = pop(&s);
        printf("PoP된 data : (%d, %d, %d) \n", item.x, item.y, item.z);
        print_stack(&s);

        printf("PoP item\n");
        item = pop(&s);
        printf("PoP data : (%d, %d, %d) \n", item.x, item.y, item.z);
        print_stack(&s);

        printf("PoP item\n");
        item = pop(&s);
        printf("PoP된 data : (%d, %d, %d) \n", item.x, item.y, item.z);
        print_stack(&s);

        ...


}
```

문제 2) 연결리스트를 이용한 큐 구현(교재 7.6절 구현)

연결리스트를 이용하여 큐을 구현하시오. 배열을 이용하여 구현한 예에서와 같이 큐 운영을 위한 기본연산인 is_empty(), is_full(), enqueue(), dequeue(), print_queue() 연산을 구현하시오.
큐에 저장되는 데이터는 아래에 정의된 3차원 세계에서의 좌표이다.

```
typedef struct WORLD_COORDINATE {
        int x;
        int y;
        int z;
}element;

typedef struct QueueNode {
        element data;
        struct QueueNode* link;
} QueueNode;

typedef struct {
        QueueNode* front, * rear;
} LinkedlistQueue;
```

is_full()은 항상 0을 반환한다. enqueue() 함수에서 스택 노드의 메모리 할당이 안될 경우 에러 처리한다.
print_queue() 함수를 수행하면 아래 실행 예와 같이
=======큐내용=======
.....
======================
형식으로 큐에 저장된 내용과 front와 rear 포인터의 위치를 표현한다.
큐가 빈 상태에서 dequeue을 할 경우 강제 종료를 하지 않고 정수 중 가장 작은 값인 {INT_MIN,
나머지 연산은 큐의 추상 데이터 형식에 저장된 함수의 기능과 동일하다.

실행 예

```c
int main(void)
{
	…
	init(&queue);			// 큐 초기화
	print_queue(&queue);

	printf("enqueue item \n");
	item.x = 1, item.y = 1, item.z = 1;
	enqueue(&queue, item);	print_queue(&queue);

	printf("enqueue item \n");
	item.x = 2, item.y = 2, item.z = 2;
	enqueue(&queue, item);	print_queue(&queue);

	printf("enqueue item \n");
	item.x = 3, item.y = 3, item.z = 3;
	enqueue(&queue, item);	print_queue(&queue);

	printf("enqueue item \n");
	item.x = 4, item.y = 4, item.z = 4;
	enqueue(&queue, item);	print_queue(&queue);

	printf("enqueue item \n");
	item.x = 5, item.y = 5, item.z = 5;
	enqueue(&queue, item);	print_queue(&queue);


	printf("dequeue item\n");
	item = dequeue(&queue);
	printf("Dequeue된 data : (%d, %d, %d) \n", item.x, item.y, item.z);
	print_queue(&queue);

	printf("dequeue item\n");
	item = dequeue(&queue);
	printf("Dequeue된 data : (%d, %d, %d) \n", item.x, item.y, item.z);
	print_queue(&queue);

	printf("dequeue item\n");
	item = dequeue(&queue);
	printf("Dequeue된 data : (%d, %d, %d) \n", item.x, item.y, item.z);
	print_queue(&queue);

	printf("dequeue item\n");
	item = dequeue(&queue);
	printf("Dequeue된 data : (%d, %d, %d) \n", item.x, item.y, item.z);
	print_queue(&queue);

	printf("dequeue item\n");
	item = dequeue(&queue);
	printf("Dequeue된 data : (%d, %d, %d) \n", item.x, item.y, item.z);
	print_queue(&queue);

	printf("dequeue item\n");
	item = dequeue(&queue);
	printf("Dequeue된 data : (%d, %d, %d) \n", item.x, item.y, item.z);
	print_queue(&queue);

	…
}
```


Microsoft Visual Studio 디버그 콘솔
========큐 내용========
< NULL > <--- Front, Rear
========================
enqueue item
========큐 내용========
<1, 1, 1> <--- Front, Rear
========================
enqueue item
========큐 내용========
<1, 1, 1> <--- Front
<2, 2, 2> <--- Rear
========================
enqueue item
========큐 내용========
<1, 1, 1> <--- Front
<2, 2, 2>
<3, 3, 3> <--- Rear
========================
enqueue item
========큐 내용========
<1, 1, 1> <--- Front
<2, 2, 2>
<3, 3, 3>
<4, 4, 4> <--- Rear
========================
enqueue item
========큐 내용========
<1, 1, 1> <--- Front
<2, 2, 2>
<3, 3, 3>
<4, 4, 4>
<5, 5, 5> <--- Rear
========================
dequeue item
Dequeue된 data : <1, 1, 1>
========큐 내용========
<2, 2, 2> <--- Front
<3, 3, 3>
<4, 4, 4>
<5, 5, 5> <--- Rear
========================
dequeue item
Dequeue된 data : <2, 2, 2>
========큐 내용========
<3, 3, 3> <--- Front
<4, 4, 4>
<5, 5, 5> <--- Rear
========================
dequeue item
Dequeue된 data : <3, 3, 3>
========큐 내용========
<4, 4, 4> <--- Front
<5, 5, 5> <--- Rear
========================
dequeue item
Dequeue된 data : <4, 4, 4>
========큐 내용========
<5, 5, 5> <--- Front, Rear
========================
dequeue item
Dequeue된 data : <5, 5, 5>
========큐 내용========
< NULL > <--- Front, Rear
========================
dequeue item
큐가 비어있음
Dequeue된 data : <-2147483648, -2147483648, -2147483648>
========큐 내용========
< NULL > <--- Front, Rear
========================
D:₩DS_Progs₩Sol_12₩Debug₩Proj.exe<프로세스 25028개>이<가> 종료되었습니다<코드: 0개>.
이 창을 닫으려면 아무 키나 누르세요...