EDDI
Electronic Design
Development Institute

에디로봇아카데미

임베디드 마스터 Lv2 과정
[자료구조 프로그래밍]

제 1기

2021. 10. 15

박태인

코드 분석

EDDI
Electronic Design Development Institute

1) 순차적 분석(구조체, node 생성)

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

typedef struct _stack stack;
struct _stack
{
        int data;
        struct _stack *link;
};
```

- _stack 이라는 구조체 타입 생성 (명칭은 stack)
- _stack 구조체 형성
  ㄴ int형 data (4 byte)
  ㄴ _stack 구조체를 가르키는 포인터 *link (8 byte)

```c
stack *create_stack_node(void)
{
        stack *tmp;

        tmp = (stack *)malloc(sizeof(stack));
        tmp->link = NULL;

        return tmp;
}
```

- 반환형 stack* 구조체 포인터
- 함수명 create_stack_node [node 추가 함수]
- Stack 포인터형 tmp 변수(구조체를 가르키는 변수 tmp)
- Tmp에 malloc을 통한 heap 영역 구조체형 메모리 생성
- 생성한 tmp 메모리의 link 값에 NULL 삽입
- 생성한 tmp 구조체형 메모리 반환

코드 분석

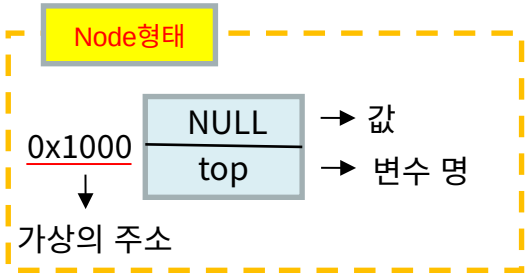
1) 순차적 분석 (main)

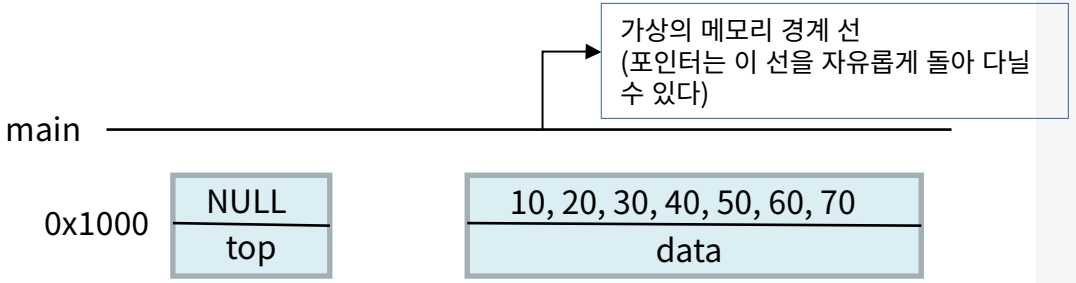
```c
int main(void)
{
        stack *top = NULL;
        int data[] = {10, 20, 30, 40, 50, 60, 70 };
        int i;

        for(i=0; i<7; i++)
        {
                push_data(&top, data[i]);
        }

         print_stack_data(top);

         for(i=0; i<8; i++)
        {
                printf("pop = %d\n", pop_data(&top));
                print_stack_data(top);
        }

        return 0;
}
```

- Main의 top변수(stack 포인터)의 값은 NULL
- Data 값은 10 ~ 70
- For문을 돌리기 위한 변수 i

- Push_data 함수
- Push_data(top의 주소 값, data[i] 번째 값

- 아래 그림 참조

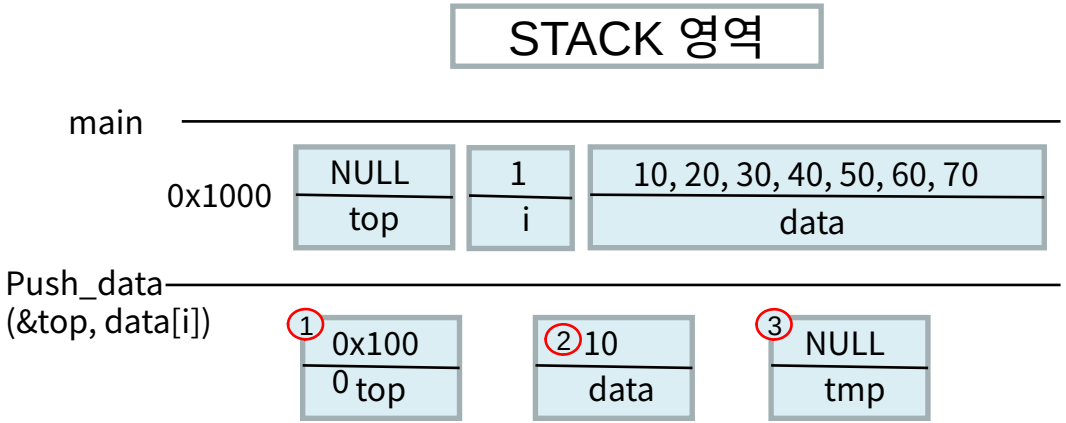




코드 분석

1) 순차적 분석 (push)

```
void push_data(stack **top, int data)
{
    stack *tmp = *top;

        *top = create_stack_node();
        (*top)->data = data;
        (*top)->link = tmp;
        return;

}
```



- Main에서 push_data(&top, data[i])로 호출
- Top, data, tmp 변수 생성 됨

① 인자로 stack* 형인 top의 주소 값을 받기 위해 stack **top으로 인자를 받음.

② Data[1]에 따른 값 10을 data 인자로 받음.

③ *top 즉, top이 가르 키는 값(0x1000 주소의 top의 값) 을 tmp에 넣음.

코드 분석

⑤ 반환된 tmp 값(0x2000)을 top이 가리키는 곳에 삽입.
ㄴ 여기서 top이 가리키는 곳은 주체권이 현재 push_data의 top에 있고(push_data 함수 실행 중 이니까) 그것의 값이 가리키는 포인팅은 main의 top 이므로 5번과 같이 변경 된다.

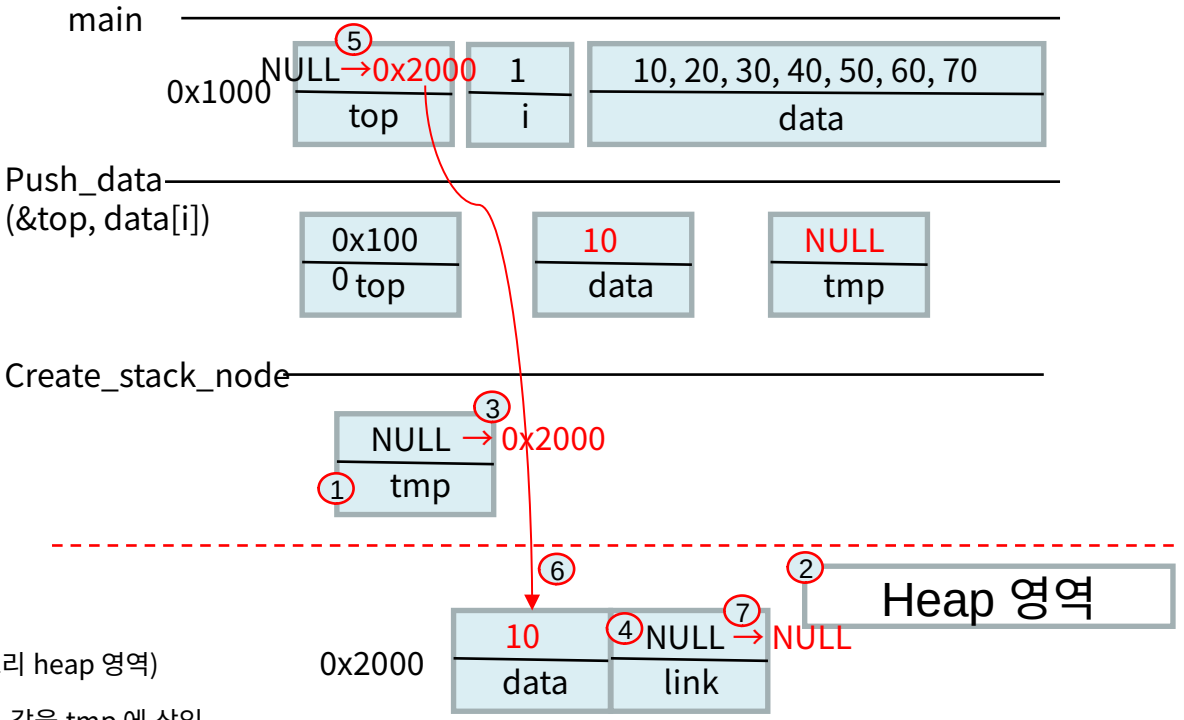
⑥ Main top이 가르키는 data 에다가 data 값을 넣는다.(현 push_data의 data 값)

⑦ Main top이 가르키는 link 에다가 tmp 값을 넣는다.(현 push_data의 tmp 값)

1) 순차적 분석 (push)

```
void push_data(stack **top, int data)
{
    1  stack *tmp = *top;

    5  *top = create_stack_node();
    6  (*top)->data = data;
    7  (*top)->link = tmp;
       return;

}
```

```
1 stack *create_stack_node(void)
{
       stack *tmp;
    3  tmp = (stack *)malloc(sizeof(stack));
    4  tmp->link = NULL;

       return tmp;
}
```



① Stack*형 tmp 변수 생성

② Malloc으로 stack 구조체 생성(이것은 메모리 heap 영역)

③ Malloc은 주소 값을 반환 한다. 반환한 주소 값을 tmp 에 삽입.

④ Tmp->link 값을 NULL로

코드 분석

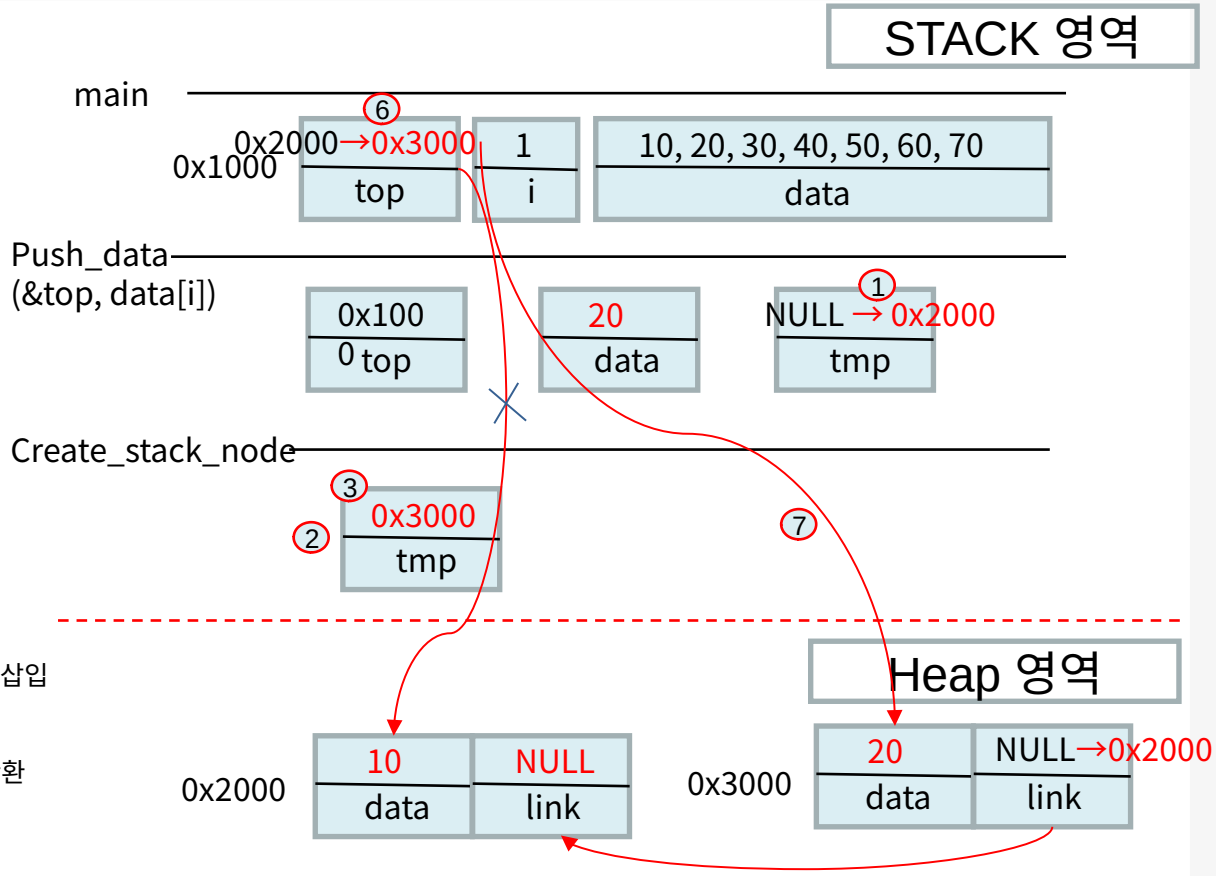
⑥ Return 받은 tmp 값을 main의 top에 삽입

⑦ Push_data의 data 값을 top이 가르키는 주소의 data에 삽입 → 메모리 끼리 연결 되는 구조가 됨

1) 순차적 분석 (두번째 push)

```
void push_data(stack **top, int data)
{
    1  stack *tmp = *top;

    6  *top = create_stack_node();
    7  (*top)->data = data;
    8  (*top)->link = tmp;
       return;

}
```

```
stack *create_stack_node(void)
{
    2  stack *tmp;
    3  tmp = (stack *)malloc(sizeof(stack));
    4  tmp->link = NULL;
    5  return tmp;
}
```



① Stack*형 tmp 변수에 main의 top의 값을 삽입

② Stack* 형 tmp 변수 생성

③ Heap 영역에 데이터 생성 및 tmp에 주소 반환

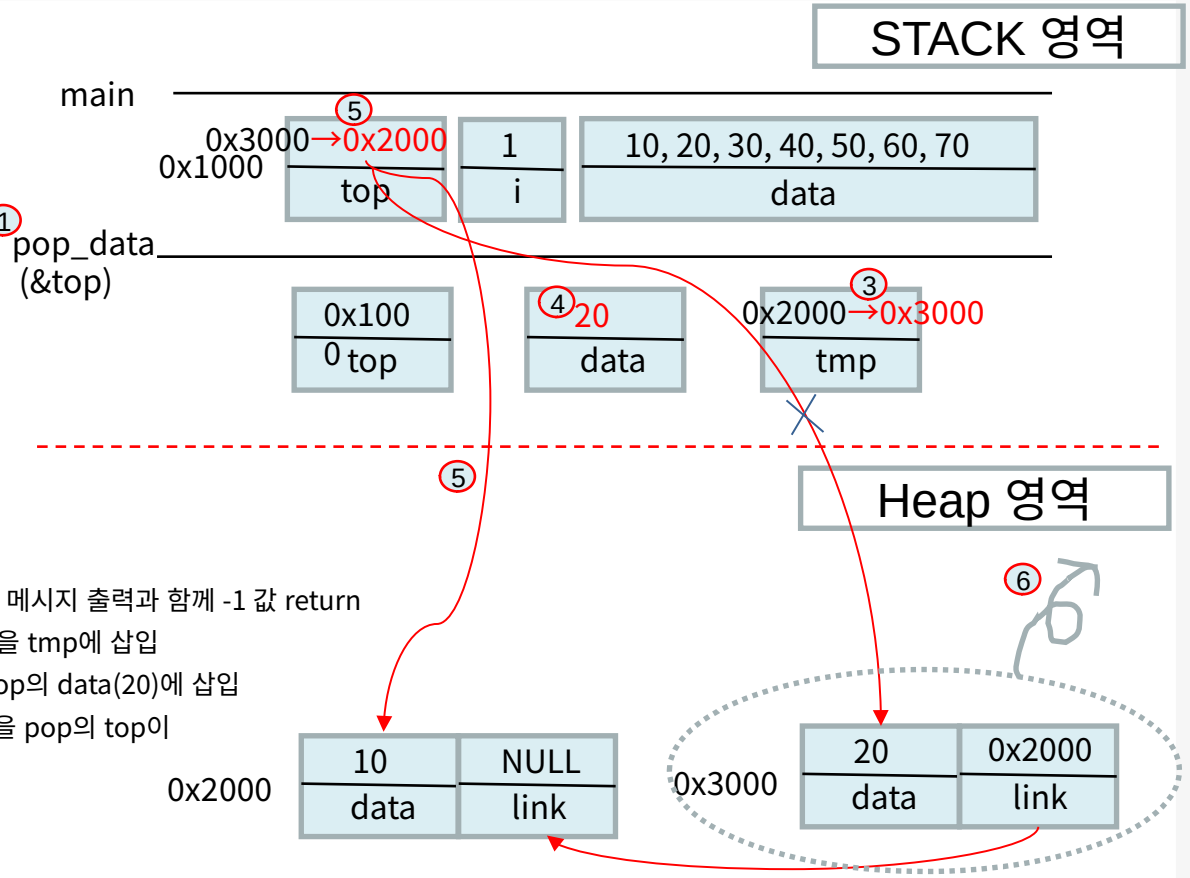
④ Tmp link에 NULL 삽입

⑤ Tmp 값 0x3000을 retrun

코드 분석

1) 순차적 분석 (pop)

```
int pop_data(stack **top)
{
    1  int data;
       stack *tmp;

    2  if(!(*top))
       {
              printf("Stack is empty\n");
              return -1;
       }

    3  tmp = *top;
    4  data = tmp->data;
    5  *top = tmp->link;
    6  free(tmp);
    7  return data;
}
```

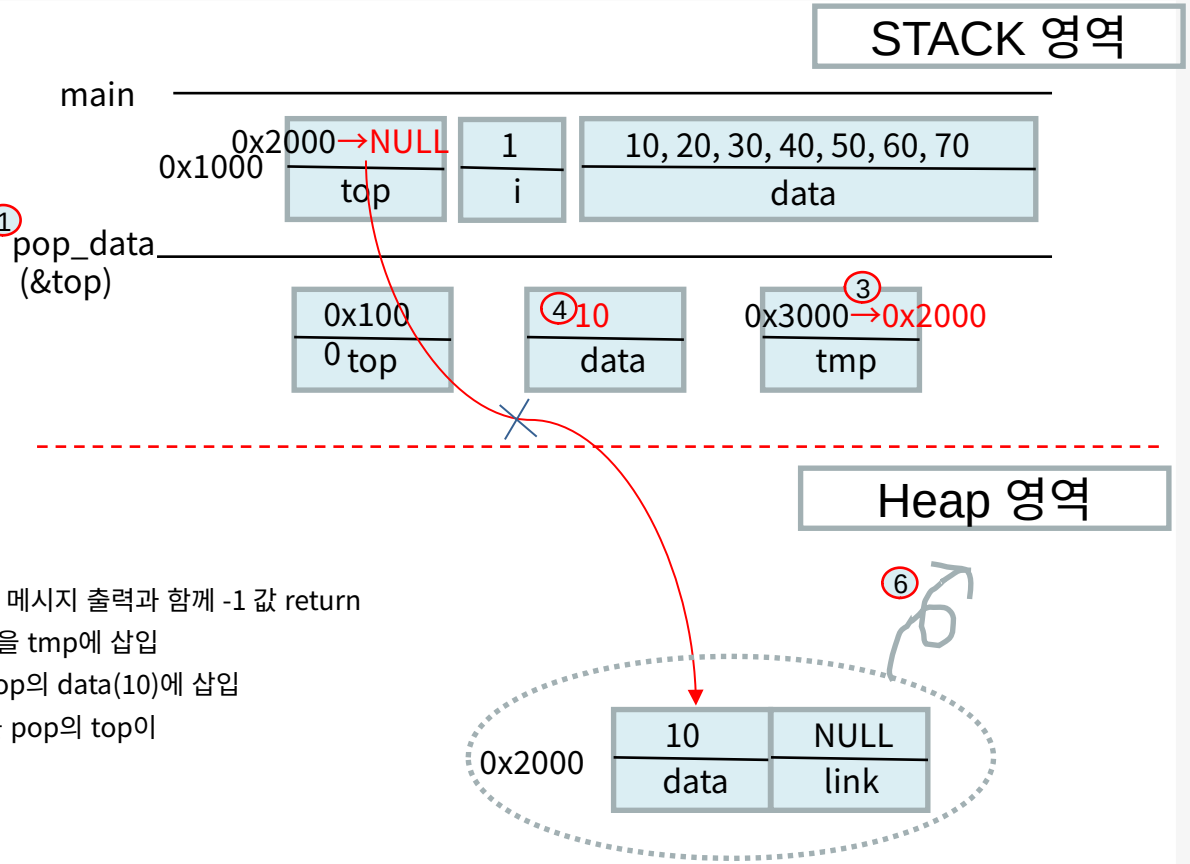


1. Top, data, tmp 변수 생성
2. 출력 할 heap 영역이 하나도 없다면 empty 메시지 출력과 함께 -1 값 return
3. Top이 가리키는 main top의 값인 0x3000을 tmp에 삽입
4. Tmp(0x3000) 가 가리키는 data 값 20을 pop의 data(20)에 삽입
5. Tmp(0x3000) 가 가리키는 link 값 0x2000을 pop의 top이 가리키는 main top에 삽입, 포인팅 변경
6. Free tmp(0x3000)
7. Data 값(20) return

코드 분석

1) 순차적 분석 (두번째 pop)

```
int pop_data(stack **top)
{
    1  int data;
       stack *tmp;

    2  if(!(*top))
       {
              printf("Stack is empty\n");
              return -1;
       }
    3  tmp = *top;

    4  data = tmp->data;
    5  *top = tmp->link;

    6  free(tmp);

    7  return data;
}
```



① Top, data, tmp 변수 생성

② 출력 할 heap 영역이 하나도 없다면 empty 메시지 출력과 함께 -1 값 return

③ Top이 가리키는 main top의 값인 0x2000을 tmp에 삽입

④ Tmp(0x2000) 가 가리키는 data 값 10을 pop의 data(10)에 삽입

⑤ Tmp(0x2000) 가 가리키는 link 값 NULL 을 pop의 top이 가리키는 main top에 삽입, 포인팅 변경

⑥ Free tmp(0x2000)

⑦ Data 값(10) return

코드 분석(전체 코드)

EDDI
Electronic Design Development Institute

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

typedef struct _stack stack;
struct _stack
{
        int data;
        struct _stack *link;
};

stack *create_stack_node(void)
{
        stack *tmp;

        tmp = (stack *)malloc(sizeof(stack));
        tmp->link = NULL;

        return tmp;
}

void push_data(stack **top, int data)
{
        stack *tmp = *top;

        *top = create_stack_node();
        (*top)->data = data;
        (*top)->link = tmp;
        return;

}


int pop_data(stack **top)
{
        int data;
        stack *tmp;

        if(!(*top))
        {
                printf("Stack is empty\n");
                return -1;
        }

        tmp = *top;

        data = tmp->data;
        *top = tmp->link;

        free(tmp);

        return data;
}
```

```c
void print_stack_data(stack *top)
{
        while(top)
        {
                printf("data = %d\n", top->data);
                top = top->link;
        }
        printf("\n");
}

int main(void)
{
        stack *top = NULL;
        int data[] = {10, 20, 30, 40, 50, 60, 70 };
        int i;

        for(i=0; i<7; i++)
        {
                push_data(&top, data[i]);
        }

         print_stack_data(top);

         for(i=0; i<8; i++)
        {
                printf("pop = %d\n", pop_data(&top));
                print_stack_data(top);
        }

        return 0;
}
```

```
data = 70
data = 60
data = 50
data = 40
data = 30
data = 20
data = 10

pop = 70
data = 60
data = 50
data = 40
data = 30
data = 20
data = 10

pop = 60
data = 50
data = 40
data = 30
data = 20
data = 10

pop = 50
data = 40
data = 30
data = 20
data = 10

pop = 40
data = 30
data = 20
data = 10

pop = 30
data = 20
data = 10

pop = 20
data = 10

pop = 10

Stack is empty
pop = -1
```

결과 값

질문

EDDI
Electronic Design Development Institute

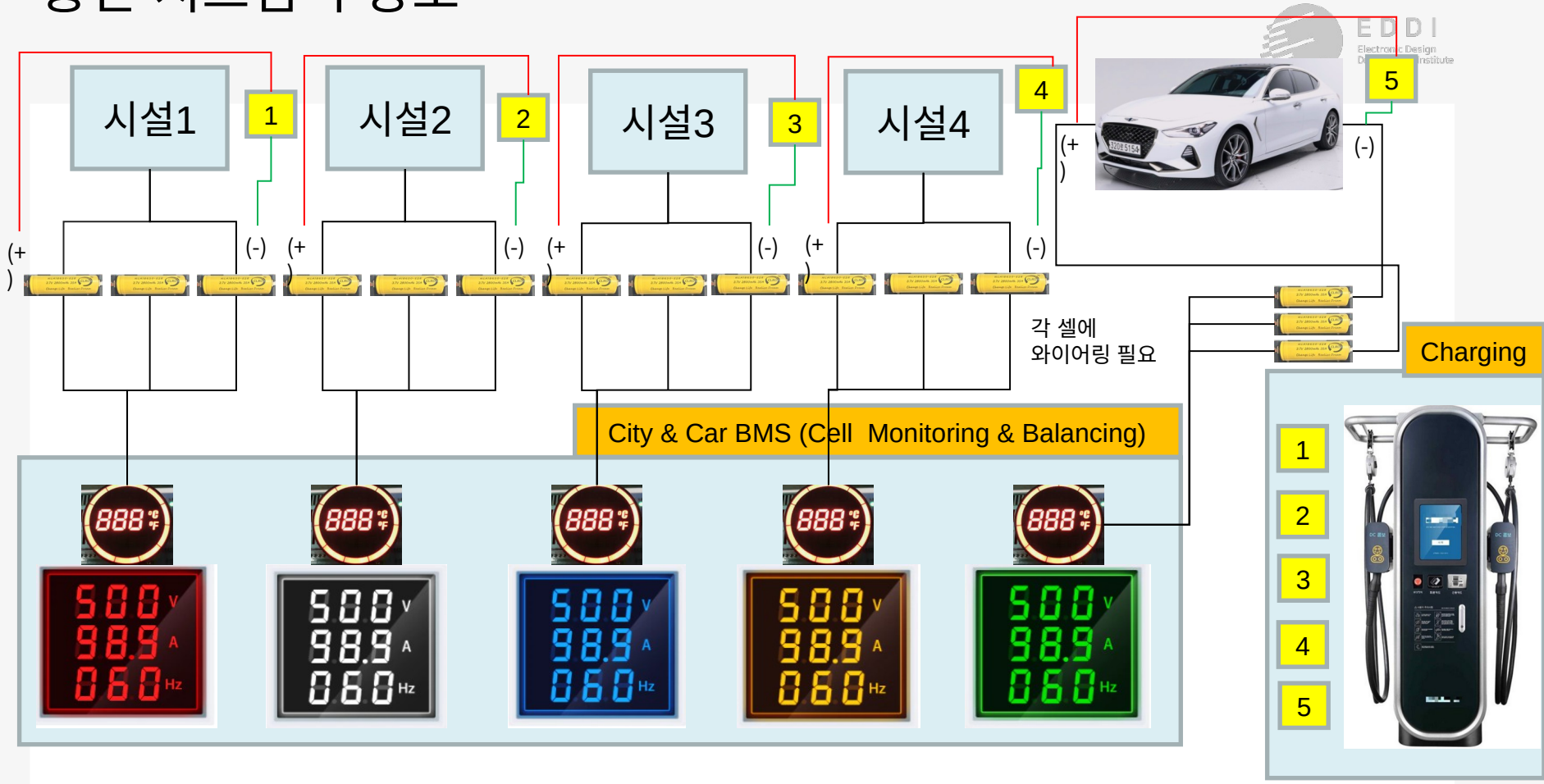
```
stack *create_stack_node(void)
{
        stack *tmp;

        tmp = (stack *)malloc(sizeof(stack));
        tmp->link = NULL;

        return tmp;
}
```

- 구조체 함수명에 포인터가 들어가는 이유가 무엇인가요?
  (그냥 반환형이 구조체 stack* 이기 때문 인 것 같기도..)

```
void push_data(stack **top, int data)
{
        stack *tmp = *top;

        *top = create_stack_node();
        (*top)->data = data;
        (*top)->link = tmp;
        return;

}
```

- Stack* 에서 top의 주소 값을 받기 위해 top이 아닌 *top이 되어 인자가 stack **top이 되는 걸까요?
  (수업 시간에 들은 것 같긴 한데, 다시 한번 문의 드립니다.)

충전 시스템 구상도


시설1
시설2
시설3
시설4
1
2
3
4
5
(+)
(-)
각 셀에
와이어링 필요
City & Car BMS (Cell Monitoring & Balancing)
Charging
500 V
98.9 A
060 Hz