18/1/24

Leetcode

Min-stack

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

int solution() {
    MinStack* obj = minStackCreate();
    minStackPush(obj, -2);
    minStackPush(obj, 0);
    minStackPush(obj, -3);

    printf("Minimum element : %d\n", minStackGetMin(obj));

    minStackPop(obj);
    printf("Top element : %d\n", minStackTop(obj));
    printf("Minimum element : %d\n", minStackGetMin(obj));

18/1/24
    minStackFree(obj);
    return 0;

typedef struct {
    int value;
    int min;
} Node;
```

```
typedef struct {
    Node *assay;
    int top;
    int size;
} Minstack;

Minstack *minstackCreate() {
    Minstack *stack = (Minstack*) malloc (sizeof(Min
    stack -> assay = (Node*) malloc (10000 * sizeof(Node
    stack -> top = -1;
    stack -> size = 10000;
    return stack;
}

void minstackPush (Minstack *obj, int val) {
    if (obj -> top == -1) {
        obj -> assay [0].value = val;
        obj -> assay [0].min = val;
        obj -> top = 0;
    } else {
        int newMin = (val < obj -> assay [obj -> top].
        val = obj -> assay [obj -> top].min;
        obj -> top++;
        obj -> assay [obj -> top].value = val;
        obj -> assay [obj -> top].min = newMin;
    }
}
```

```
void minStackPop(MinStack *obj) {
    if (obj->top != -1) {
        return obj->array[obj->top].value;
    }
    return -1;
}
int minStackGetMin(MinStack *obj) {
    if (obj->top != -1) {
        return obj->array[obj->top].min;
    }
    return -1;
}
void minStackFree(MinStack *obj) {
    free(obj->array);
    free(obj);
}
int solution() {
    MinStack *obj = minStackCreate();
    minStackPush(obj, -2);
    minStackPush(obj, 0);
    minStackPush(obj, -3);

    printf("Min element : %d\n", minStackGetMin(obj));
```

Input :-

[-2] [0] [-3]

Output :-

-3, 0, -2