LeetCode

-1번, 7번-

안태진(taejin@codecure.smuc.ac.kr)

상명대학교 보안동아리 Code Cure

목 차

LeetCode 1번

- 정수의 배열(nums)과 배열의 크기(numsSize)와 값(target)이 함수의 인자로 전달됨, 배열의 두 정수를 더했을 때 특정한 값(target)이 되는 정수를 찾고 그 인덱스를 반환하는 함수 작성
  - 특별한 메소드와 기능이 필요하지 않고, 제일 익숙한 프로그램이 C언어라 C언어 선택

```c
int * twoSum(int *nums, int numsSize, int target) {
    int *result = malloc(sizeof(int) * 2);

    for (int i = 0; i < numsSize; i++) {
        for (int j = i + 1; j < numsSize; j++) {
            if (nums[i] + nums[j] == target) {
                result[0] = i;
                result[1] = j;

                return result;
            }
        }
    }

    return NULL;
}
```

Accepted	52 ms

LeetCode 1번

```
int * twoSum(int *nums, int numsSize, int target) {
    int *result = malloc(sizeof(int) * 2);

    for (int i = 0; i < numsSize; i++) {
```

- int *result = malloc(sizeof(int) * 2);
  - 두 정수의 인덱스 값을 저장할 배열 동적 할당
- for (int i = 0; i < numsSize; i++)
  - i는 작은 인덱스 값, 그러므로 0부터 시작, numsSize – 1까지 증가 하며 반복

LeetCode 1번

```
        for (int j = i + 1; j < numsSize; j++) {
            if (nums[i] + nums[j] == target) {
                result[0] = i;
                result[1] = j;

                return result;
```

- for (int j = i + 1; j < numsSize; j++)
  - j는 큰 인덱스 값, 그러므로 i + 1부터 numsSize – 1까지 증가 하며 반복
- if (nums[i] + nums[j] == target)
  - 두개의 정수 값과 주어진 값이 같은지 비교
  - 같다면 resul에 i와 j 할당하고 nums 반환

LeetCode 7번

- 32bit의 부호있는 정수가 인자로 전달되면 그 정수를 거꾸로 뒤집어 정수로 다시 반환하는 함수 작성
  - Python3 : 파이썬의 .append 메소드와 list를 쓰고, 제곱을 표현하기 위해 선택

```python
def reverse(self, x):
    ''' type x : int
    rtype : int '''
    rx = 0
    str1 = str(x)
    rstr1 = list()
    lastIndex = len(str1) - 1
    sign = False

    for i in range(lastIndex, -1, -1):
        rstr1.append(str1[i])

    lastIndex = len(rstr1) - 1
    if (rstr1[lastIndex] == '-'):
        sign = True
        del rstr1[lastIndex]
        lastIndex = len(rstr1) - 1

    for i in range(len(rstr1)):
        rx = rx + int(rstr1[i]) * (10 ** (lastIndex - i))
    if (sign == True):
        rx = rx * -1


    if((rx > 2**31 - 1) or (rx < -2 ** 31)):
        return 0
    else:
        return rx
```

Accepted	96 ms	python3

LeetCode 7번

```
    rx = 0
    str1 = str(x)
    rstr1 = list()
    lastIndex = len(str1) - 1
    sign = False
```

- rx = 0: 인자로 받은 x의 리버스값
- str1 = str(x): 인자 x를 인덱스 값으로 접근하기 위해 문자열로 변환
- rstr1 = list(): str1을 거꾸로 저장하기 위해 list로 선언
- lastIndex = len(str1) – 1: 마지막 인덱스 값을 많이 사용하기 때문에, 따로 선언
- sign = False: 부호가 있는지 없는지에 따라 코드 흐름 달라짐, 디폴트값 false

LeetCode 7번

```
for i in range(lastIndex, -1, -1):
    rstr1.append(str1[i])
```

- rstr1 만들기
  - range(lastIndex, -1, -1) : 마지막 인덱스부터 0까지 -1씩 감소시킴
  - rstr1.append(str1[i]) : 뒤의 값 부터 하나씩 rstr1에 추가

LeetCode 7번

```
    lastIndex = len(rstr1) - 1
    if (rstr1[lastIndex] == '-'):
        sign = True
        del rstr1[lastIndex]
        lastIndex = len(rstr1) - 1
```

- 부호 처리
  - lastIndex = len(rstr1) – 1: lastIndex를 rstr1의 마지막 인덱스로 변경
  - 만약 음수라면 마지막 인덱스에 ‘-’이 들어감
  - 따라서 부호사인 true로 변경, 마지막 인덱스 삭제, lastIndex 다시 계산

LeetCode 7번

```
for i in range(len(rstr1)):
    rx = rx + int(rstr1[i]) * (10 ** (lastIndex - i))
if (sign == True):
    rx = rx * -1
```

- rx 만들기
  - rstr1의 각 자리는 10^(lastIndex – i)의 자리
    - e.g., rstr1 = [‘3’, ‘2’, ‘1’] -> (3 * 10 ^ 2) + (2 * 10 ^ 1) + (1 * 10 ^ 0) = 321
  - 만약 부호가 있다면 rx에 -1곱해줘 음수로 변경

LeetCode 7번

```
if((rx > 2**31 - 1) or (rx < -2 ** 31)):
    return 0
else:
    return rx
```

- 오버플로우 검사 & 반환
  - 32비트의 구간 -2^31 <= rx <= 2^ 31 – 1을 넘어가면 32비트가 아니기 때문에 0반환
  - 아니라면 rx 반환

감사합니다!