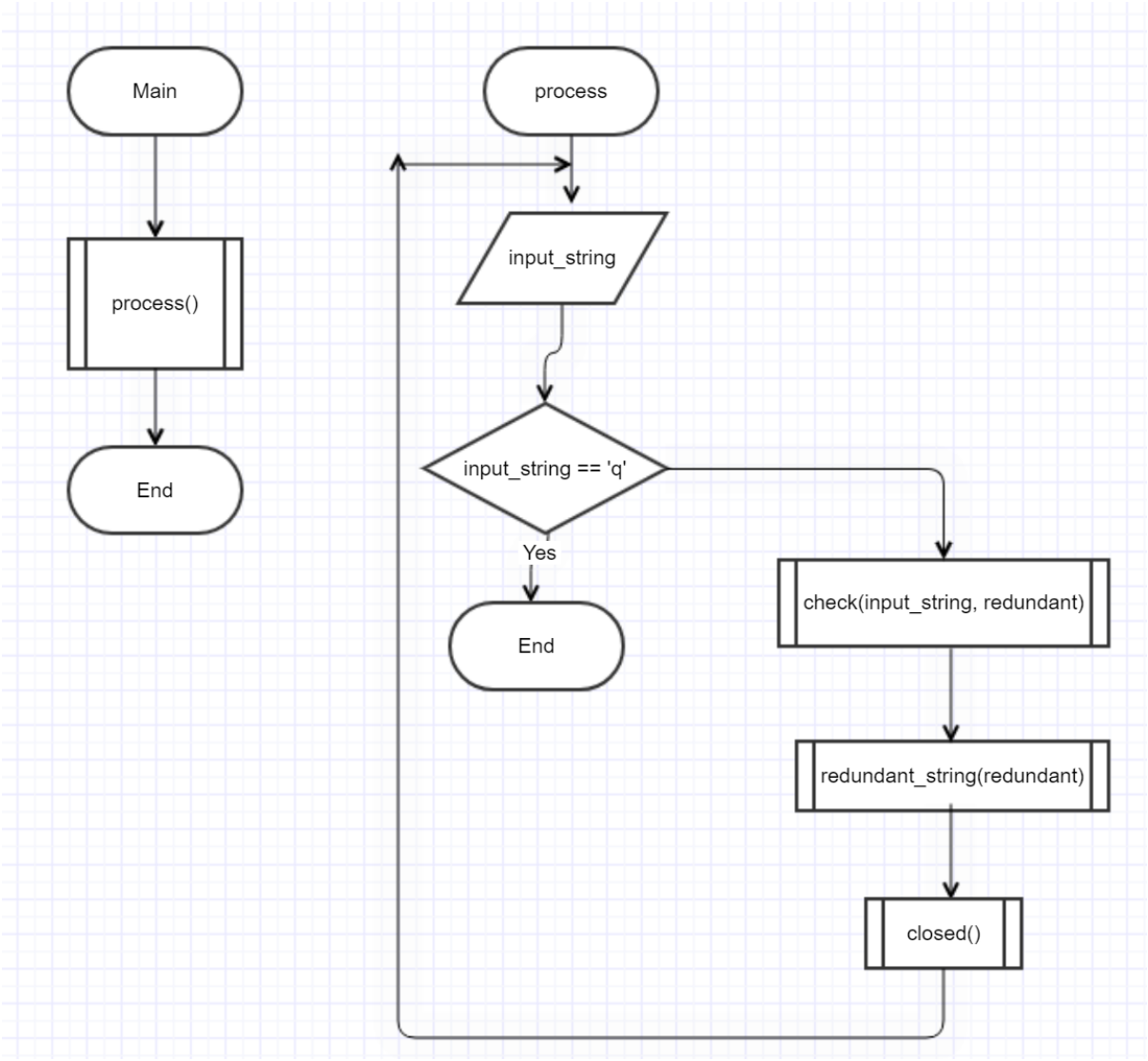
<자료구조 3차 프로그래밍 보고서 – Stack>

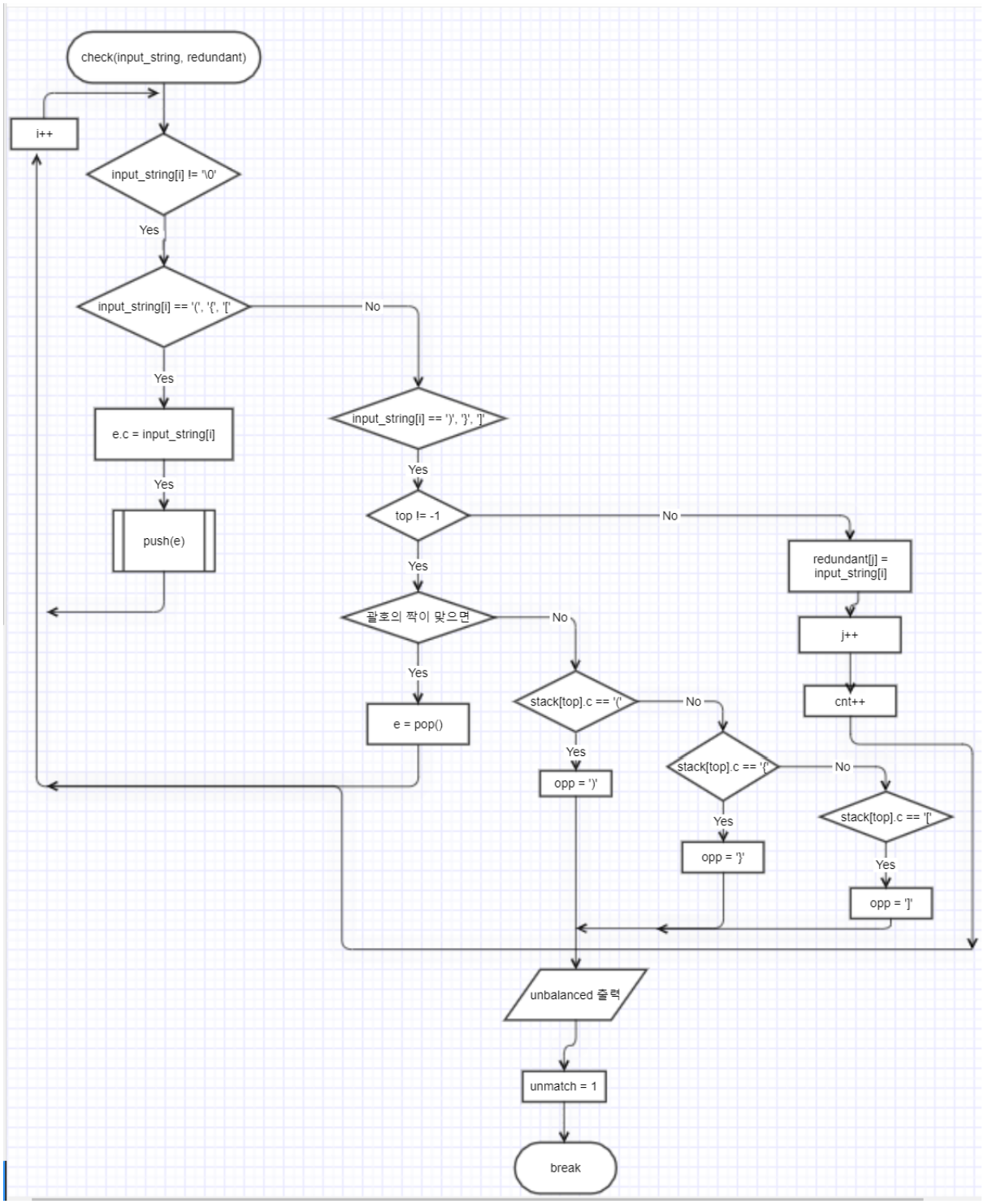
소프트웨어학과 201720736 이균

1. Flowchart

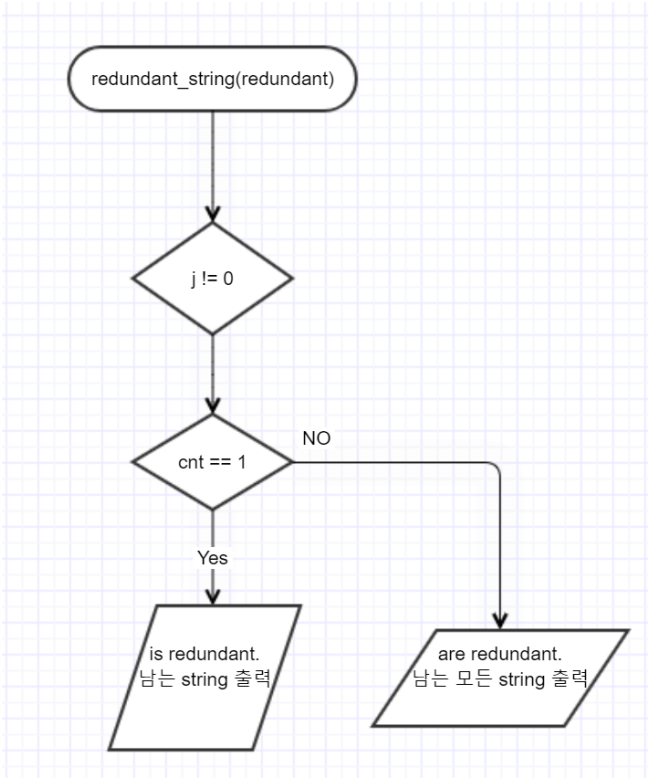


main에서 process()를 호출한다.

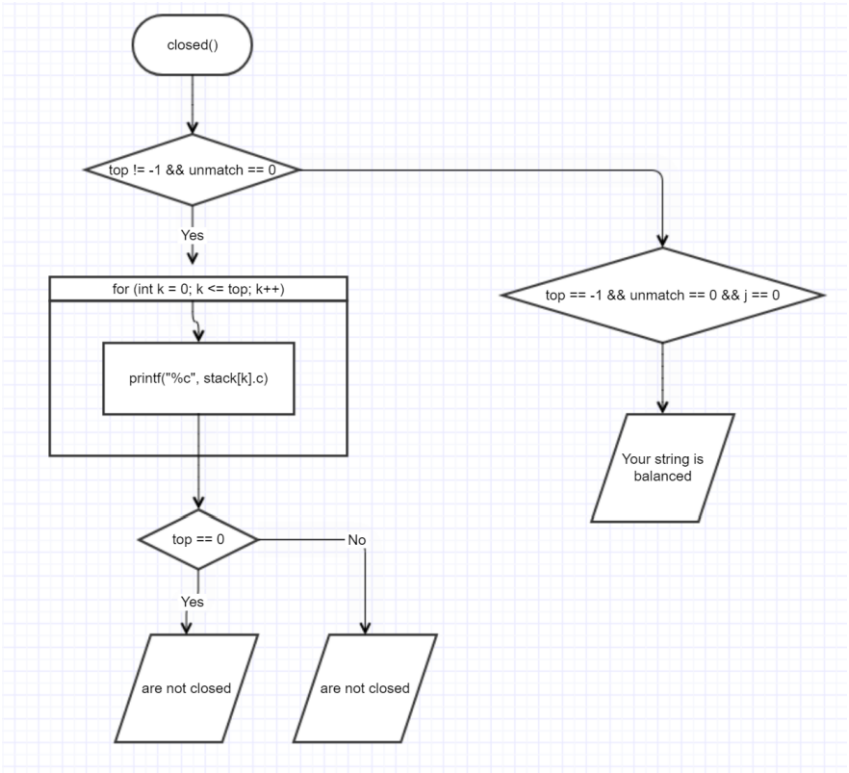
Process()는 string을 입력 받고 string이 q가 아니면 계속 반복한다. 또한 check, redundant_string, closed함수를 순차적으로 호출한다.



문자열이 끝날 때 까지 문자열에서 여는 괄호는 push하여 stack에 집어넣고 짝이 맞는 닫는 괄호이면 pop을 한다. 이때 닫는 괄호이지만 짝이 안 맞는 경우 알맞은 짝으로 바꾸어 출력한다. 또한 stack에 아무것도 없는데 닫는 괄호만 덩그러니 나온 경우는 redundant한 경우이기 때문에 새로운 배열에 남는 괄호를 넣어준다.



Redundant한 case인 경우 redundant 배열에 있는 string을 출력한다. 이때 하나 이상이면 are, 하나면 is라고 출력형식을 바꿔준다.



여는 괄호는 앞서 stack에 넣기 때문에 stack에 여는 괄호가 남아있는 경우는 not closed인 상황이다.

2. Data structure

String이 입력으로 들어온다. 이때 string에 괄호가 포함되는데 괄호는 여는 괄호와 닫는 괄호가 있다. 주어진 문제는 이 괄호들이 제대로 닫혔는지(balanced) 아니면 닫히지 않은 것(not closed)이 있는지, 짝이 맞지 않는 괄호가 들어왔는지(not balanced), 닫히는 괄호만 덩그러니 남겨져 있는지(redundant) 판별하는 문제이다.

따라서 Data structure로 Stack을 이용하여 여는 괄호가 들어올 경우 stack에 push한 다음, 짝에 맞는 닫히는 괄호가 들어온 경우 pop을 하면 된다.

문자열의 끝까지 짝이 맞는 괄호들을 push, pop을 한 뒤에 stack에 남는 괄호가 있을 경우 not closed인 것이다.

반면 여는 괄호를 stack에 push한 뒤에 짝이 맞지 않는 닫는 괄호가 온다면 짝이 맞게 괄호를 바꿔주고 괄호를 알맞게 바꾸어 준 뒤 출력하면 될 것이다.

또한 닫히는 괄호가 뒤에 남아있는 redundant상황인 경우 맨 마지막에 남아있는 닫히는 괄호들을 새로운 배열에 저장하면 된다.

만약 redundant하지도 않고 not balanced도 아니고 짝이 다 맞아서 stack에 아무것도 없을 경우는 balanced인 경우일 것이다.

```
void push(element item)
{
    if (top >= MAX_STACK_SIZE - 1)
    {
        stackFull();
        return;
    }
    stack[++top] = item;
}

element pop()
{
    if (top == -1)
    {
        return stackEmpty();
    }
    return stack[top--];
}
```

다음은 Stack의 push, pop에대한 코드이다.

Stack은 top이라는 변수를 사용하여 stack에 아이템을 넣고 top을 증가시킨다. 또한 stack에 있는 아이템을 꺼낼 때는 top을 감소시킨다.