substring reverse를 실행할 때, substring의 시작 지점과 끝 지점을 나타내는 s\_e라는 구조체를 정의하였습니다. 그리고, s\_e의 값을 출력하는 print\_s\_e라는 함수도 같이 정의하였습니다.

처음에는 s1를 s2로 만드는 방법을 사용하였지만, 결과를 출력하는 형식이 원하는 형태로 나오지 않아서(역순으로 출력됨.) s2를 s1을 만드는 방법을 구한 후에 출력하는 방법을 채택했습니다.

main함수에서 문자열 2개(s1, s2)를 입력 받은 후, find\_sol함수가 작동합니다.

find\_sol 설명

1. s1과 s2의 길이가 다르다면 “no solution”을 출력하고 return을 실행해서 함수를 종료합니다.

2. substring reverse를 실행하기 위한 char stack(string)과 뒤집는 과정을 기록하기 위한 s\_e stack(s\_es)을 초기화합니다.

3. idx(인덱스)와 len(문자열의 길이)의 길이를 초기화합니다. 그리고 문자열 s2를 복제하여 문자열 s3만듭니다.

4. i(=idx)가 len보다 커질 때까지 다음 내용을 반복합니다

4-1. new(s\_e)를 선언합니다.

4-2. i(=idx)부터 len까지 다음 내용를 반복합니다.

4-2-1. s1과 s3가 같다면, s\_es에 저장되어 있는 substring reverse의 과정들을 출력하고, return을 실행해서 함수를 종료합니다.

4-2-2. check라는 문자를 선언하고, s3의 i번째 문자를 저장합니다. 그후 string stack에 check를 push합니다.

4-2-3. 그 후, string stack의 top을 이용해서 check에 string의 첫 번째 문자를 저장합니다.

4-2-4. idx번째의 s1과 check가 같고, idx와 i가 다른 경우,

4-2-4-1. new.start에 idx를 new.end에 i를 저장하고, new를 s\_es에 push합니다.

4-2-4-2. substring reverse를 실행합니다. (j를 new.start부터 new.end까지 반복하여 string stack의 pop을 이용해서 s3의 j 번째 문자를 저장합니다.) 그리고 break를 사용해서 4-2의 반복문을 멈추고 idx에 1을 더하고 4-3로 갑니다.

4-2-5 그리고 idx번째의 s1과 check가 같지만, idx와 i가 다른 경우엔 break를 사용해서 4-2의 반복문을 멈추고 4-3으로 갑니다.

4-3. idx에 1을 더하고 4-1로 갑니다.

5. 4.의 반복문을 빠져나왔다면 정답을 못 찾은 경우이기 때문에, “no solution”을 출력하고 함수를 종료합니다.

When executing substring reverse, a struct called s\_e is defined to indicate the start and end points of the substring, and a function called print\_s\_e is also defined to print the value of s\_e.

Initially, the method used was to convert s1 to s2, but the output format did not come out as desired (it was reversed). So I adopted the method of converting s2 to s1.

In the main function, two strings are entered in s1 and s2, and the find\_sol function operates.

find\_sol function:

1. If the lengths of s1 and s2 are different, print "no solution" and exit the function using return.

2. Initialize a char stack (string) for executing substring reverse and a stack (s\_es) for recording the reversal process.

3. Initialize the index (idx) and the length of the string (len). Then, copy s2 to create s3.

4. Repeat the following steps until i (equal to idx) is greater than len:

4-1. Declare a s\_e (new).

4-2. Repeat the following steps from i (put idx) to len:

4-2-1. If s1 and s3 are the same, print the sequence of substring reverse stored in s\_es and exit the function using return.

4-2-2. Declare a character(check) and store the i-th character of s3 in check. Then, push check to the string stack.

4-2-3. Then, use top function of string stack to store the first character of string stack in check.

4-2-4. If the idx-th character of s1 is the same as check and idx and i are different:

4-2-4-1. Store idx in new.start and i in new.end, then push new to s\_es.

4-2-4-2. Execute substring reverse (repeat from j= new.start to new.end, and use pop of the string stack to store the j-th character of s3). Then, use break to stop the loop in step 4-2 and go to 4-3, adding.

4-2-5. If the idx-th character of s1 is the same as check but idx and i are different, use break to stop the loop in step 4-2 and go to 4-3.

4-3. Add 1 to idx and go to 4-1.

5. If we exit the loop in step 4 without finding a solution, print "no solution" and exit the function.

Therefore, when you enter two strings in this program, it prints sequence of substring reverse operations that convert 's1' to 's2'.