4.1 스택이란?

스택 후입선출 Last In First Out

가장 최근에 들어온 것이 가장 위에 위치, 먼저 나감

스택상단 스택에서 입출력이 이루어지는 부분

스택하단 스택의 바닥부분

요소 스택에 저장되는 것

공백스택 스택에 요소가 하나도 없을 때

예제:시스템 스택을 이용한 함수 호출 함수호출 실행이 끝나면 자신을 호출한 함수로 되돌아감 🡪 스택 사용

스택 복귀할 주소를 기억하는 데 사용됨

시스템스택 운영체제가 사용하는 스택

활성레코드 함수가 저장될 때마다 활성레코드가 만들어짐 🡨 복귀 주소 저장됨

프로그램 카운터, 매개변수, 지역변수가 같이 저장됨

추상 자료형 스택 정의 0개 이상의 요소를 가지는 선형리스트의 일종

연산 스택에 요소 추가∙삭제∙검사

ADT1.1 객체: 0개 이상의 원소를 가지는 유한 선형 리스트

연산:

create(size) :: = 최대 크기가 size인 공백 스택

is\_full(s) :: = //스택의 포화상태 검사

if (스택의 원소수 == size) return TRUE;

else return FALSE;

is\_empty(s) :: = //스택의 공백상태 검사

if (스택의 원소수 == 0) return TRUE;

else return FALSE;

push(s, item):: = //삽입연산

if (is\_full(s)) return ERROR\_STACKFULL;

else 스택의 맨 위에 item을 추가한다

pop(s) ::= //제거연산

if (is\_empty(s)) return ERROR\_STACKEMPTY;

else 스택의 맨 위에 원소를 제거해서 반환한다

peek(s)::=

if (is\_empty(s)) return ERROR\_STACKEMPTY;

else 스택의 맨 위에 원소를 제거하지 않고 반환한다

Quiz 1. 후입선출

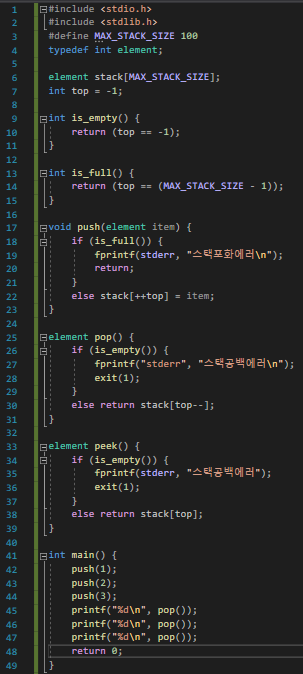
2. 입력: adecb

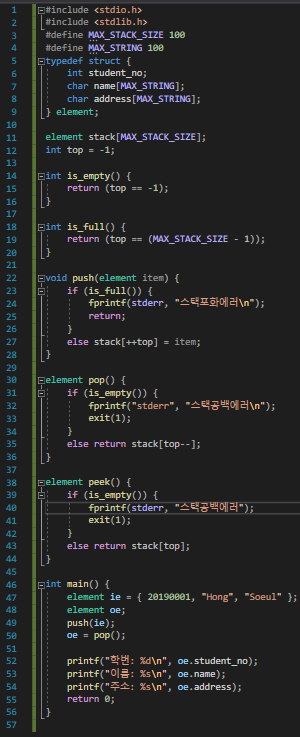
출력: bceda

4.2 스택의 구현

요소를 저장하는 1차원 배열

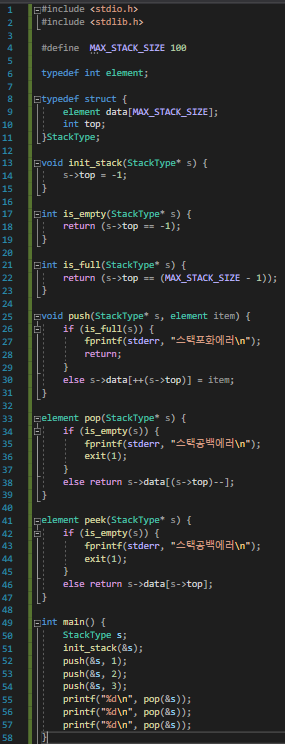
가장 최근에 입력되었던 자료를 가리키는 top변수 스택이 비어있으면 -1의 값

전역변수로 구현 프로그램4.1 

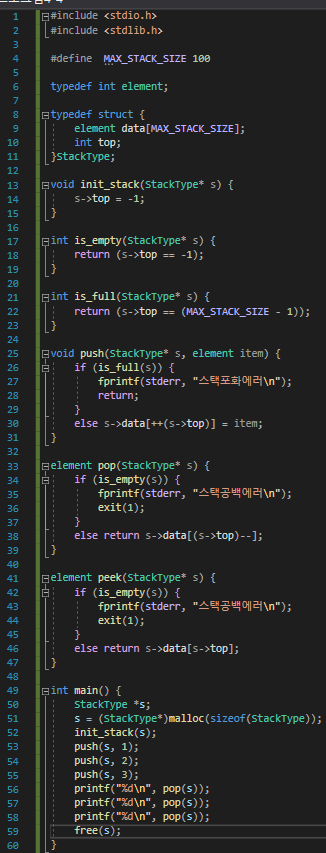
스택의 요소를 구조체로 하기 프로그램4.2 

관련된 데이터를 함수의 매개변수로 전달하는 방법 스택을 전역변수로 선언 = 동시에 여러 개의 스택 사용이 어려움

top, stack을 하나의 구조체로 결합, 구조체의 포인터를 함수에 전달

프로그램4.3 

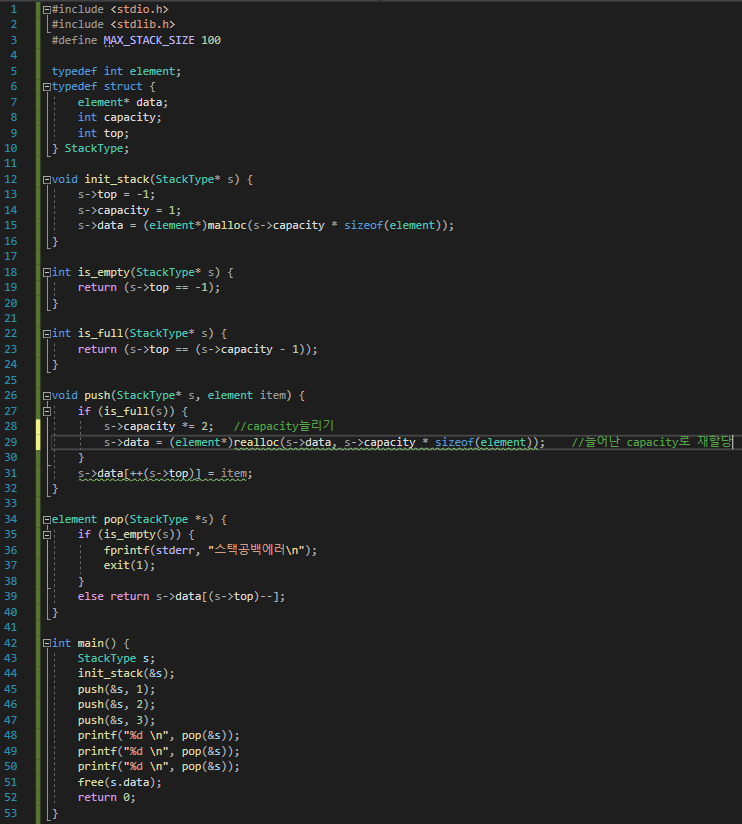
도전문제 123

스택을 동적 메모리 할당으로 생성하는 방법 프로그램4.4 

4.3 동적배열스택

실행시간에 메모리를 할당받기 malloc()사용

필요할 때마다 스택의 크기를 동적으로늘릴 수 있음

프로그램4.5 

4.4 스택의 응용: 괄호 검사 문제

괄호의 검사 조건 조건1 왼쪽 괄호의 개수와 오른쪽 괄호의 개수가 같아야한다

조건2 같은 종류의 괄호에서 왼쪽 괄호는 오른쪽 괄호보다 먼저 나와야 한다

조건3 서로 다른 종류의 왼쪽 괄호와 오른쪽 괄호 쌍은 서로를 교차하면 안된다

알고리즘 왼쪽 괄호를 만나면 스택에 삽입, 오른쪽 괄호를 만나면 스택에서 괄호를 꺼낸후 오른쪽 괄호와 짝이 맞는 지 검사

알고리즘4.5 check\_matching(expr):

while(입력 expr의 끝이 아니면)

ch <- expr의 다음 글자

switch(ch)

case '(': case '[': case'{':

ch를 스택에 삽입

break

case ')': case ']': case'}':

if(스택이 비어있으면)

then 오류

else 스택에서 open\_ch를 꺼낸다

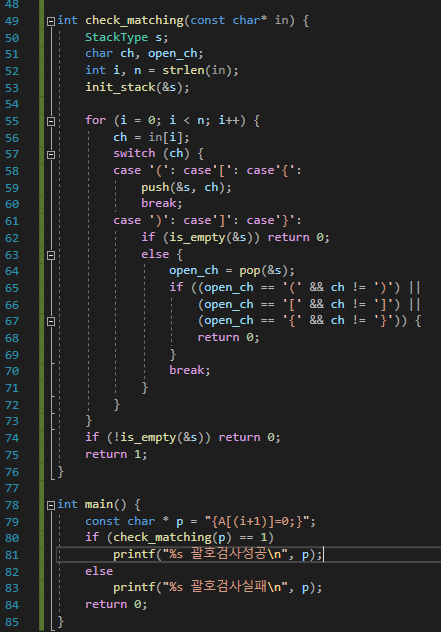
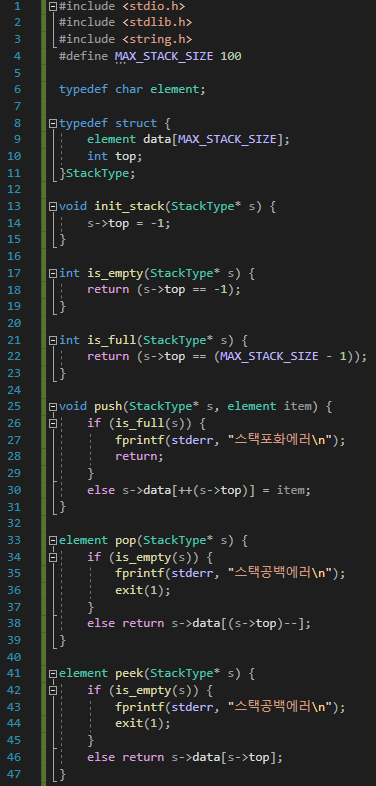
if(ch와 open\_ch가 같은 짝이아니면)

then 오류 보고

break

if(스택이 비어 있지 않으면)

then 오류보고

프로그램4.6 

Quiz 1. {( 🡪 {[ --> 0

{[( 🡪 { 🡪 오류발생

4.5 스택의 응용: 후위 표기 수식의 계산

중위연산, 후위연산, 전위연산

중위연산 연산자가 피연산자 사이에 있음 ex) 2+3\*4

후위연산 연산자가 피연산자 뒤에 있음 ex) +2\*34

전위연산 연산자가 피연산자 앞에 있음 ex) 234\*+

컴파일러는 후위표기법으로 변환한 뒤 스택을 이용해 계산 컴파일러가 후위표기법을 선호하는 이유 우선 계산할 내용을 표시

수식을 읽으면서 바로 계산 가능

스택을 이용한 후위표기 수식 계산 피연산자 스택 삽입

연산자 연산 시행 후, 결과를 스택에 저장

스택에 원하는 만큼의 피연산자가 없으면 오류 발생

알고리즘4.6 calc\_posfix:

스택s를 생성하고 초기화한다

for item in 후위초기식 do

if (item이 피연산자이면)

push(s, item)

else if(item이 연산자 op이면)

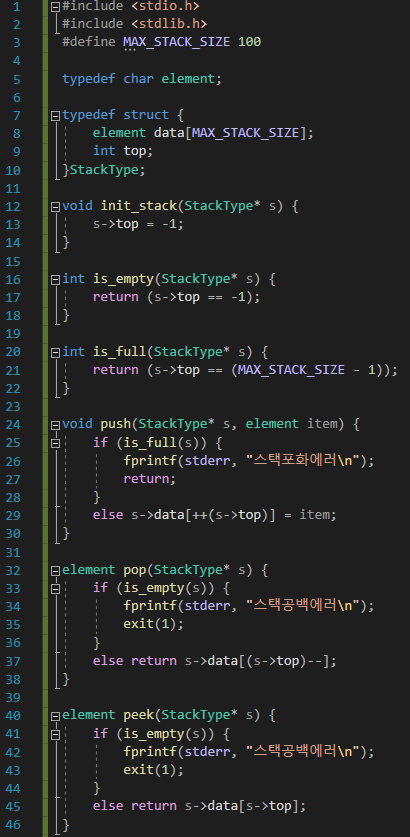
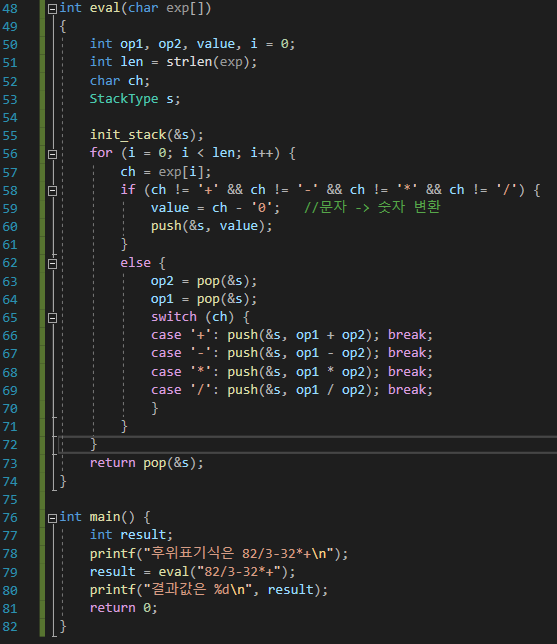
second <-- pop(s)

first<--pop(s)

result <-- first op second

push(s, result)

final\_result <-- pop(s)

프로그램4-7  

중위표기수식을 후위표기수식으로 변환 프로그래머: 중위표기, 컴파일러: 후위표기 🡪 변환필요

피연산자 피연사자를 바로 후위표기 수식에 출력

연산자 연산자 우선순위, 순서에 따라 다름

알고리즘4.7 infix\_to\_postfix(exp):

스택s를 생성하고 초기화

while(exp에 처리할 문자가 남아 있으면):

ch <-- 다음에 처리할 문자

switch(ch)

case 연산자:

while (peek(s)의 우선순위 >= ch의 우선순위) do

e < --pop(s)

e를 출력

push(s, ch);

break;

case 왼쪽괄호:

push(s, ch);

break;

case 오른쪽괄호:

e <-- pop(s)

while(e!=왼쪽괄호)do

e를 출력

e <-- pop(s)

break;

case 피연산자:

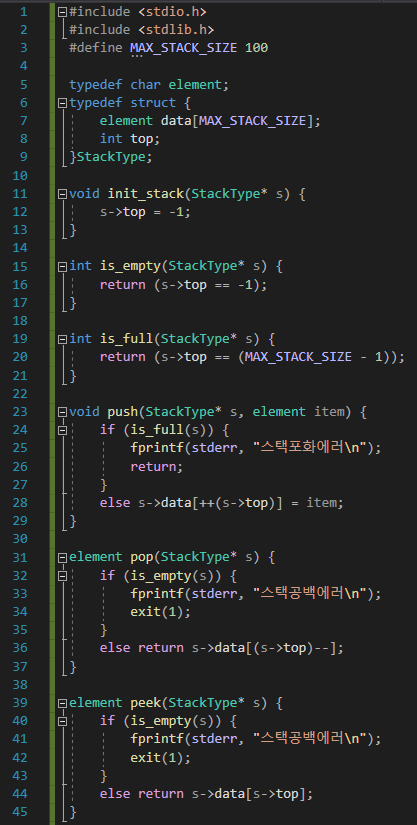
ch를 출력

break;

while(not is\_empty(s)) do

e <-- pop(s)

e를 출력

프로그램4.8 

Quiz \* 🡪 \*( 🡪 \*(+ 🡪 \*% 🡪 \* 🡪

4.6 스택의 응용: 미로문제

출구를 찾는 기본적인 방법 경로 선택 🡪 시도 🡪 실패 🡪 다른 경로 재시도

다른 경로 저장필요

최근에 저장한 경로가 쉽게 추출되는 자료구조 = 스택

풀이 현재 위치에서 이동이 가능한 칸의 위치를 스택에 저장

스택에서 맨위의 위치를 꺼내어 현재의 위치로 한 다음, 같은 작업 반복

한번 거쳐간 위치를 다시 검사하지 않도록 해 무한루프 방지

알고리즘4.8 maze\_search() :

스택s와 출구의 위치x, 현재 생쥐의 위치를 초기화

while (현재의 위치가 출구가 아니면) do

현재의 위치를 방문한 것으로 표기

if (현재위치의 위, 아래, 왼쪽, 오른쪽 위치가 아직 방문되지 않았고 갈 수 있으면)

then 그 위치들을 스택에 push

if (is\_empty(s))

then 실패

else 스택에서 하나의 위치를 꺼내어 현재의 위치로 만든다;

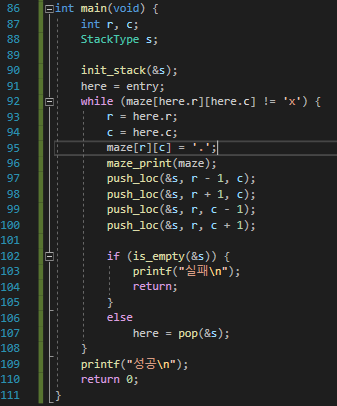
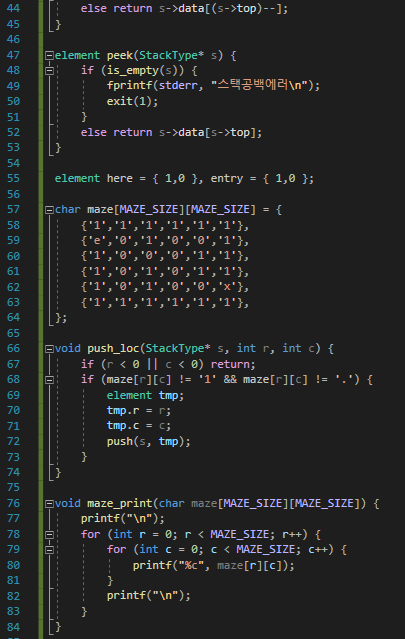
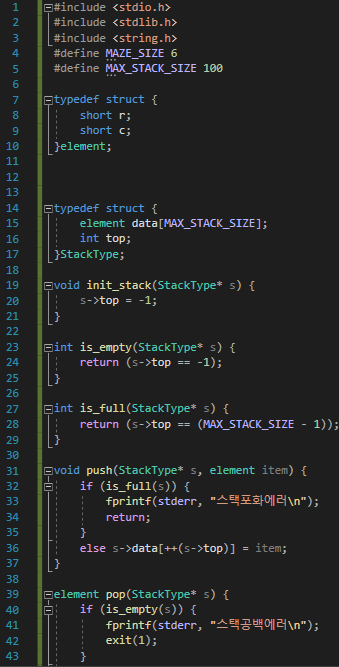
성공

2차원 배열을 통해 미로 구현 0 갈 수 있는 길

1. 갈 수 없는 벽

m 쥐

. 방문이 끝난 위치

프로그램4.9 

Quiz {1,1} 🡪 {1,2} 🡪 {1,3} 🡪 {2,2}, {1,3} 🡪 {2,2} 🡪{3,2} 🡪 {4,2} 🡪 {4,1}{4,3} 🡪{4,1}{4,4} 🡪 {4,1}{4,5} 🡪골인

연습문제

1. 4

2. 2

3. 10, 20

4. 4

5. 1,3

6. 1

7. 1

8.

push(A, 1); //1

push(A, 2); //12

push(A, 3); //123

pop(A); //12

push(A, 4); //124

push(A, 5); //1245

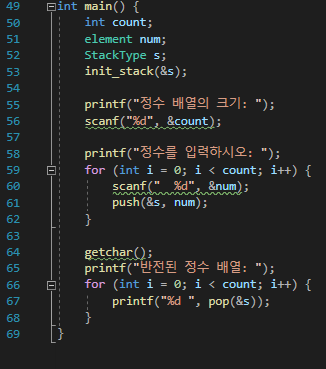
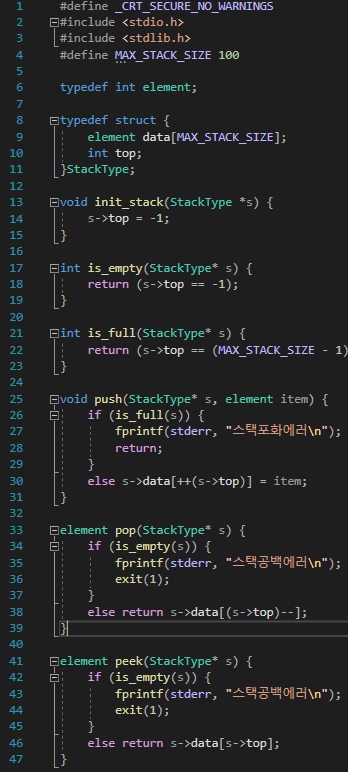
pop(A); //124

9.

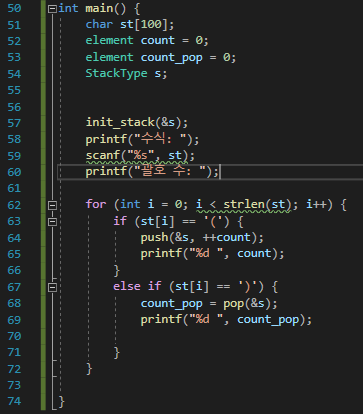
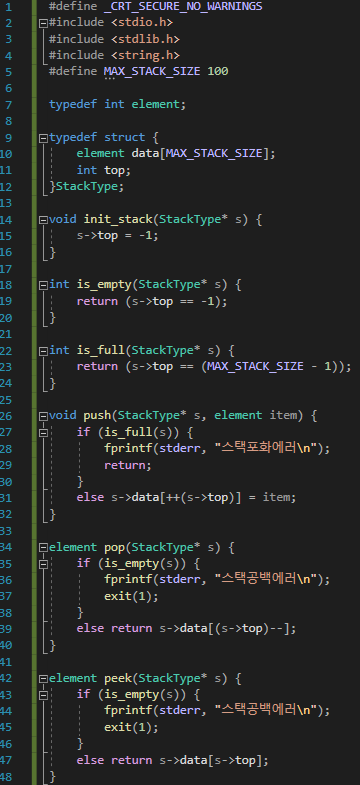
A: abc

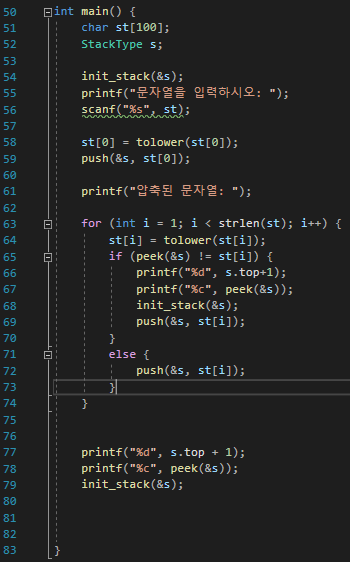
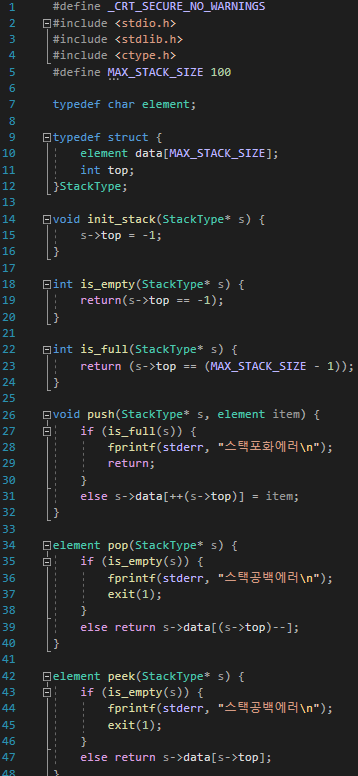
B:

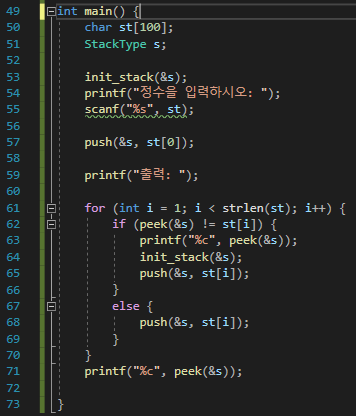
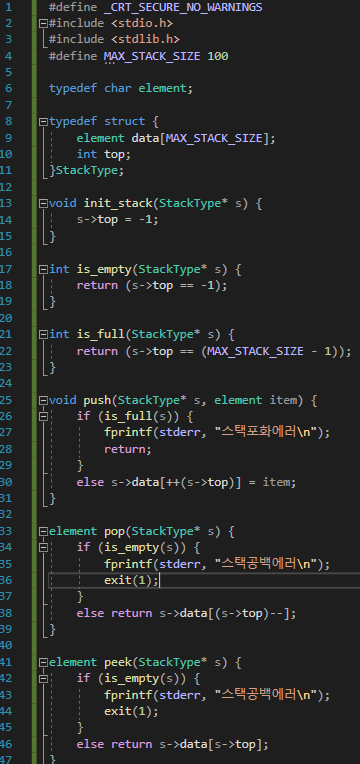
10.

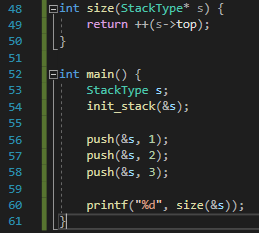
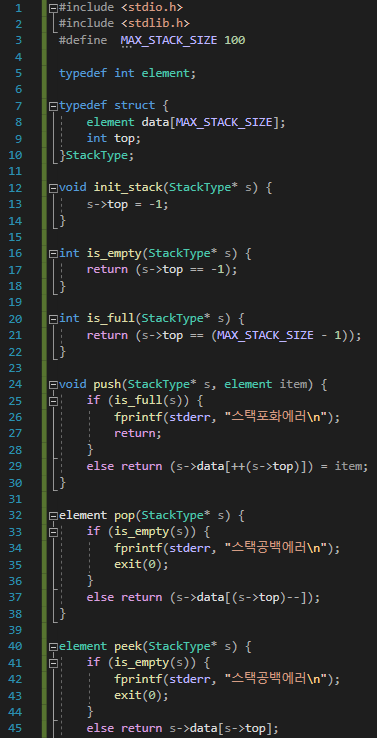


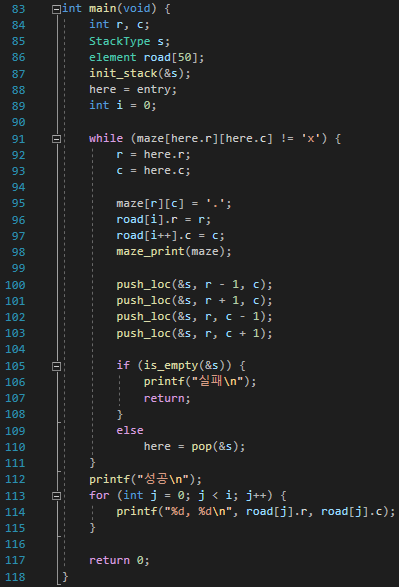
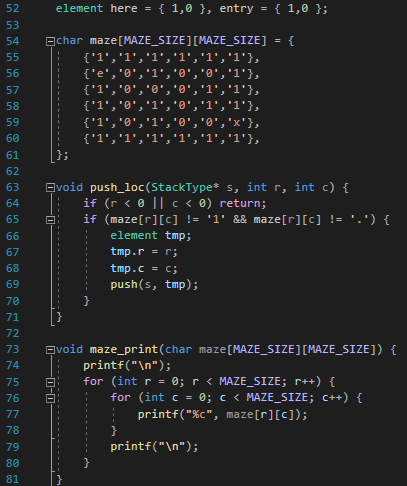
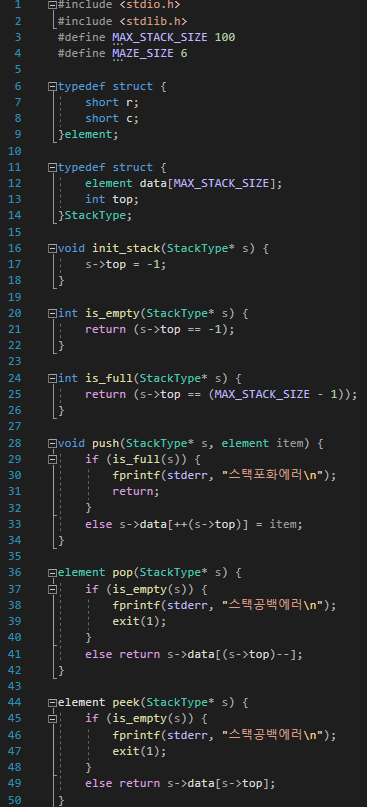
11.



12. 

13. 

14. 

15. 

16.

