# 01. 스택에서 삽입 작업이 발생하면 top의 값은 어떻게 변경되는가?

(1) top = 0 (2) top = 1  
 (3) top = top-1 (4) top = top+1

삽입 작업이 발생될 때 우선 top의 값을 증가시키고 값을 저장하는 순서를 거친다. 그러므로, 스택 안의 요소 개수가 늘어나면 top의 값이 늘어난다.

# 02. 문자 A, B, C, D, E를 스택에 넣었다가 다시 꺼내어 출력하면 어떻게 되는가?

(1) A, B, C, D, E (2) E, D, C, B, A  
 (3) A, B, C, E, D (4) B, A, C, D, E

스택은 Last-In-First-Out 구조이기 때문에 넣은 역순으로 꺼내게 된다.

# 03. 10, 20, 30, 40, 50을 스택에 넣었다가 3개의 항목을 삭제하였다. 남아있는 항목은?

{10, 20}

스택은 LIFO 구조이기 때문에 최근 3개의 항목인 50, 40, 30이 삭제된다.

# 04. 배열로 구현된 스택에서 top가 3이면 현재 스택에 저장된 요소들의 개수는?

(1) 1 (2) 2  
 (3) 3 (4) 4

배열의 인덱스 0부터 3까지 요소가 저장되어 있을 때 top의 값이 3이다.

# 05. 다음 중 배열로 구현된 스택에서 공백상태에 해당하는 조건은? 또 포화상태에 해당되는 조건은?

(1) top == -1 (2) top == 0  
 (3) top == (MAX\_STACK\_SIZE-1) (4) top == MAX\_STACK\_SIZE

스택이 공백상태일 때는 top == -1이고 포화상태일 때는 top == (MAX\_STACK\_SIZE-1)이다. 스택은 인덱스 0부터 시작이기에 공백은 -1이여야 하고 포화일 때는 사전에 정의가 된 최대 스택 사이즈보다 작은 MAX\_STACK\_SIZE - 1이다.

# 06. 스택에 항목들을 삽입하고 삭제하는 연산은 시간 복잡도가 어떻게 되는가?

(1) O(1) (2) O(log₂n)  
 (3) O(n) (4) O(n²)

스택의 모든 연산은 O(1)의 시간 복잡도를 가진다.

# 07. 다음은 어떤 수식의 후위 표기이다. 이때 최초로 수행되는 연산은 어느 것인가?

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| A | B | E | + | D | \* | – |

(1) B + E (2) E + A  
 (3) D \* B (4) B \* E

중위 표기법으로 나타내면 A - (B + E) \* D 이다.

# 08. 크기가 5인, 배열로 구현된 스택 A에 다음과 같이 삽입과 삭제가 되풀이되었을 경우에 각 단계에서 스택의 내용(1차원 배열의 내용, top의 값)을 나타내시오.

push(A, 1);  
 push(A, 2);  
 push(A, 3);  
 pop(A);  
 push(A, 4);  
 push(A, 5);  
 pop(A);

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | A |  | A |  | A |  | A |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  | top --> | 3 |  |  |
|  |  | top --> | 2 |  | 2 | top --> | 2 |
| top --> | 1 |  | 1 |  | 1 |  | 1 |
|  | A |  | A |  | A |  |  |
|  |  |  |  |  |  |  |  |
|  |  | top --> | 5 |  |  |  |  |
| top --> | 4 |  | 4 | top --> | 4 |  |  |
|  | 2 |  | 2 |  | 2 |  |  |
|  | 1 |  | 1 |  | 1 |  |  |

# 09. A와 B가 스택이라고 하고, a, b, c, d가 객체라고 하자. 다음의 일련의 스택 연산을 수행한 뒤의 각각의 스택을 그려라.

push(A, a);  
 push(A, b);  
 push(A, c);  
 push(B, d);  
 push(B, pop(A));  
 push(A, pop(b));  
 pop(B);

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | A |  | B |  |
|  |  |  |  |
|  |  |  |  |
| top --> | c |  |  |
|  | b |  |  |
|  | a |  |  |

# 10. 배열에 들어있는 정수의 순서를 거꾸로 하는 프로그램을 작성해보자. 스택을 사용한다.

실행결과

정수 배열의 크기: 6  
정수를 입력하시요: 1 2 3 4 5 6  
반전된 정수 배열: 6 5 4 3 2 1

#include <stdio.h>  
#include <stdlib.h>  
#define MAX\_STACK\_SIZE 100  
  
typedef int element;  
typedef struct {  
 element data[MAX\_STACK\_SIZE];  
 int top;  
} StackType;  
  
// 스택 초기화 함수  
void init\_stack(StackType \*s) {  
 s->top = -1;  
}  
  
// 공백 검사 함수  
int is\_empty(StackType \*s) {  
 return (s->top == -1);  
}  
// 포화 검사 함수  
int is\_full(StackType \*s) {  
 return (s->top == (MAX\_STACK\_SIZE - 1);  
}  
// 삽입함수  
void push(StackType \*s, element item) {  
 if (is\_full(s)) {  
 fprintf(stderr, “스택 포화 에러\n”);  
 return;  
 }  
 else s->data[++(s->top)] = item;  
}  
// 삭제함수  
element pop(StackType \*s) {  
 if (is\_empty(s)) {  
 fprintf(stderr, “스택 공백 에러\n”);  
 exit(1);  
 }  
 else return s->data[(s->top)--];  
}  
  
int main() {  
 int len, num;  
 StackType s;  
 init\_stack(&s);  
  
 printf(“정수 배열의 크기: “);  
 scanf(“%d”, &len);  
  
 printf(“정수를 입력하시오: “);  
 for (int i = 0; i < len; i++) {  
 scanf(“%d”, &num);  
 push(&s, num);  
 }  
  
 printf(“반전된 정수 배열: “);  
 for (int i = 0; i < (len-1); i++) {  
 printf(“%d “, pop(&s));  
 }  
 printf(“%d”, pop(&s));  
  
 return 0;  
}

# 11. 수식에 있는 괄호의 번호를 출력하는 프로그램을 작성하라. 왼쪽 괄호가 나올 때마다 괄호 번호는 하나씩 증가한다. 오른쪽 괄호가 나오면 매칭되는 왼쪽 괄호 번호를 출력한다.

실행결과

수식: ( ( ( ) ) ( ( ) ) )  
괄호 수: 1 2 3 3 2 4 5 5 4 1

수식: ( ( ( ( ( ( ( )  
괄호 수: 1 2 3 4 5 5

#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
#define MAX\_STACK\_SIZE 100  
  
typedef char element;  
typedef struct {  
 element data[MAX\_STACK\_SIZE];  
} StackType;  
  
void init\_stack(StackType \*s) {  
 s->top = -1;  
}  
  
// 공백 검사 함수  
int is\_empty(StackType \*s) {  
 return (s->top == -1);  
}  
// 포화 검사 함수  
int is\_full(StackType \*s) {  
 return (s->top == (MAX\_STACK\_SIZE - 1));  
}  
// 삽입함수  
void push(StackType \*s, element item) {  
 if (is\_full(s)) {  
 fprintf(stderr, “스택 포화 에러\n”);  
 return;  
 }  
 else s->data[++(s->top)] = item;  
}  
// 삭제함수  
element pop(StackType \*s) {  
 if (is\_empty(s)) {  
 fprintf(stderr, “스택 공백 에러\n”);  
 exit(1);  
 }  
 else return s->data[(s->top)--];  
}  
  
int main() {  
 int len, num = 0;  
 StackType s;  
 init\_stack(&s);  
  
 char str[MAX\_STACK\_SIZE] = {};  
 printf(“수식: “);  
 scanf\_s(“%s”, str, sizeof(str));  
 len = strlen(str);  
  
 printf(“괄호 수: “);  
 for (int i = 0; i < len; i++) {  
 if (str[i] == ‘(‘) {  
 push(&s, ++num);  
 printf(“%d “, num);  
 }  
 else if (str[i] == ‘)’) {  
 printf(“%d “, pop(&s));  
 }  
 else exit(1);  
 }  
 return 0;  
}

# 12. 다음과 같이 문자열을 압축하는 프로그램을 작성하라. “4a3b”는 ‘a’가 4개, ‘b’가 3개 있다는 의미이다. 이러한 압축 방법을 런길이(run length) 압축이라고 한다. 소문자와 대문자는 구별하지 않는다. 압축된 문자열에서는 소문자로 출력한다. 스택의 peek() 연산을 고려해보자.

실행결과

문자열을 입력하시오: aaaAbBb  
압축된 문자열: 4a3b

#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
#define MAX\_STACK\_SIZE 100  
  
typedef char element;  
typedef struct {  
 element data[MAX\_STACK\_SIZE];  
 int top;  
} StackType;  
  
// 스택 초기화 함수  
void init\_stack(StackType \*s) {  
 s->top = -1;  
}  
  
// 공백 검사 함수  
int is\_empty(StackType \*s) {  
 return (s->top == -1);  
}  
// 포화 검사 함수  
int is\_full(StackType \*s) {  
 return (s->top == (MAX\_STACK\_SIZE – 1));  
}  
// 삽입함수  
void push(StackType \*s, element item) {  
 if (is\_full(s)) {  
 fprintf(stderr, “스택 포화 에러\n”);  
 return;  
 }  
 else s->data[++(s->top)] = item;  
}  
// 삭제함수  
element pop(StackType \*s) {  
 if (is\_empty(s)) {  
 fprintf(stderr, “스택 공백 에러\n”);  
 exit(1);  
 }  
 else return s->data[(s->top)--];  
}  
// 피크함수  
element peek(StackType \*s) {  
 if (is\_empty(s)) {  
 fprintf(stderr, “스택 공백 에러\n”);  
 exit(1);  
 }  
 else return s->data[s->top];  
}  
  
int main() {  
 int len, countA, countB;  
 countA = 0;  
 countB = 0;  
 StackType s;  
 init\_stack(&s);  
  
 char string[MAX\_STACK\_SIZE];  
 printf(“문자열을 입력하시오: “);  
 scanf\_s(“%s”, string, sizeof(string));  
 len = strlen(string);  
  
 printf(“압축된 문자열: “);  
 for (int i = 0; i < len; i++) {  
 if ((string[i] == ‘A’) || (string[i] == ‘a’)) {  
 countA++;  
 } else if ((string[i] == ‘B’) || (string[i] == ‘b’)) {  
 countB++;  
 } else exit(1);  
 }  
 printf(“%da%db", countA, countB);  
 return 0;  
}

# 13. 주어진 정수에서 반복되는 숫자를 제거하는 프로그램을 작성해보자. 스택 사용을 고려해보자.

실행결과

정수를 입력하시오: 122233  
출력: 123

#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
#define MAX\_STACK\_SIZE 100  
  
typedef char element;  
typedef struct {  
 element data[MAX\_STACK\_SIZE];  
 int top;  
} StackType;  
  
// 스택 초기화 함수  
void init\_stack(StackType \*s) {  
 s->top = -1;  
}  
  
// 공백검사 함수  
int is\_empty(StackType \*s) {  
 return (s->top == -1);  
}  
// 포화검사 함수  
int is\_full(StackType \*s) {  
 return (s->top == (MAX\_STACK\_SIZE – 1));  
}  
// 삽입함수  
void push(StackType \*s, element item) {  
 if (is\_full(s)) {  
 fprintf(stderr, “스택 포화 에러\n”);  
 return;  
 }  
 else s->data[++(s->top)] = item;  
}  
// 삭제함수  
element pop(StackType \*s) {  
 if (is\_empty(s)) {  
 fprintf(stderr, “스택 공백 에러\n”);  
 exit(1);  
 }  
 else return s->data[(s->top)--];  
}  
// 피크함수  
element peek(StackType \*s) {  
 if (is\_empty(s)) {  
 fprintf(stderr, “스택 공백 에러\n”);  
 exit(1);  
 }  
 else return s->data[s->top];  
}  
  
int main() {  
 StackType s;  
 init\_stack(&s);  
 int len;  
  
 element string[MAX\_STACK\_SIZE];  
 printf(“정수를 입력하시오: “);  
 scanf\_s(“%s”, string, sizeof(string));  
 len = strlen(string);  
  
 printf(“출력: “);  
 printf(“%s”, string[0]);  
 for (int i = 1; i < len; i++) {  
 if (string[i] != string[i-1]) {  
 printf(string[i]);  
 i++;  
 }  
 else i++;  
 }  
 return 0;  
}

# 14. 배열로 구현된 스택에 저장된 요소의 수를 반환하는 size 연산을 구현하여 보라.

element size(StackType \*s) {  
 return s->top + 1;  
}

# 15. 미로 탐색 프로그램에서 탐색 성공 시에 입구부터 출구까지의 경로를 출력하도록 수정하라.

#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
#define MAZE\_SIZE 6  
#define MAX\_STACK\_SIZE 100  
  
typedef struct {  
 short r;  
 short c;  
} element;  
  
typedef struct {  
 element data[MAX\_STACK\_SIZE];  
 int top;  
} StackType;  
  
// 스택 초기화 함수  
void init\_stack(StackType \*s) {  
 s->top = -1;  
}  
  
// 공백검사 함수  
int is\_empty(StackType \*s) {  
 return (s->top == -1);  
}  
// 포화검사 함수  
int is\_full(StackType \*s) {  
 return (s->top == (MAX\_STACK\_SIZE – 1));  
}  
// 삽입함수  
void push(StackType \*s, element item) {  
 if (is\_full(s)) {  
 fprintf(stderr, “스택 포화 에러\n”);  
 return;  
 }  
 else s->data[++(s->top)] = item;  
}  
// 삭제함수  
element pop(StackType \*s) {  
 if (is\_empty(s)) {  
 fprintf(stderr, “스택 공백 에러\n”);  
 exit(1);  
 }  
 else return s->data[(s->top)--];  
}  
// 피크함수  
element peek(StackType \*s) {  
 if (is\_empty(s)) {  
 fprintf(stderr, “스택 공백 에러\n”);  
 exit(1);  
 }  
 else return s->data[s->top];  
}  
element here = {1, 0}, entry = {1, 0};  
char maze[MAZE\_SIZE][MAZE\_SIZE] = {  
 { ‘1’, ‘1’, ‘1’, ‘1’, ‘1’, ‘1’ },  
 { ‘e’, ‘0’, ‘1’, ‘0’, ‘0’, ‘1’ },  
 { ‘1’, ‘0’, ‘0’, ‘0’, ‘1’, ‘1’ },  
 { ‘1’, ‘0’, ‘1’, ‘0’, ‘1’, ‘1’ },  
 { ‘1’, ‘0’, ‘1’, ‘0’, ‘0’, ‘x’ },  
 { ‘1’, ‘1’, ‘1’, ‘1’, ‘1’, ‘1’ },  
};  
// 위치를 스택에 삽입  
void push\_loc(StackType \*s, int r, int c) {  
 if (r < 0 || c < 0) return;  
 if (maze[r][c] != ‘1’ && maze[r][c] != ‘.’) {  
 element tmp;  
 tmp.r = r;  
 tmp.c = c;  
 push(s, tmp);  
 }  
}  
  
// 미로를 화면에 출력한다.  
void maze\_print(char maze[MAZE\_SIZE][MAZE\_SIZE]) {  
 printf(“\n”);  
 for (int r = 0; r < MAZE\_SIZE; r++) {  
 for (int c = 0; c < MAZE\_SIZE; c++) {  
 printf(“%c”, maze[r][c]);  
 }  
 printf(“\n”);  
 }  
}  
  
int main(void) {  
 int r, c;  
 StackType s;  
 element path[MAX\_STACK\_SIZE];  
  
 init\_stack(&s);  
 here = entry;  
 int index = 0;  
 while (maze[here.r][here.c] != ‘x’) {  
 r = here.r;  
 c = here.c;  
 maze[r][c] = ‘.’;  
 path[index].r = r;  
 path[index++].c = c;  
 maze\_print(maze);  
 push\_loc(&s, r – 1, c);  
 push\_loc(&s, r + 1, c);  
 push\_loc(&s, r, c – 1);  
 push\_loc(&s, r, c + 1);  
 if (is\_empty(&s)) {  
 printf(“실패\n”);  
 return;  
 }  
 else  
 here = pop(&s);  
 }  
 printf(“성공\n”);  
 printf(“경로:\n”);  
 for (int i = 0; i < index; i++) {  
 printf(“%d %d\n”, path[i].r, path[i].c);  
 }  
 return 0;  
}

# 16. 회문(palindrome)이란 앞뒤 어느 쪽에서 읽어도 같은 단어를 의미한다. 예를 들면 “eye”, “madam, I’m Adam”, “race car” 등이다. 여기서 물론 구두점이나 스페이스, 대소문자 등은 무시하여야 한다. 스택을 이용하여 주어진 문자열이 회문인지 아닌지를 결정하는 프로그램을 작성하라.

실행결과

문자열을 입력하시오: madam  
회문입니다.

#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
#include <ctype.h>  
#define MAX\_STACK\_SIZE 100  
  
typedef char element;  
typedef struct {  
 element data[MAX\_STACK\_SIZE];  
 int top;  
} StackType;  
  
// 스택 초기화 함수  
void init\_stack(StackType \*s) {  
 s->top = -1;  
}  
  
// 공백검사 함수  
int is\_empty(StackType \*s) {  
 return (s->top == -1);  
}  
// 포화검사 함수  
int is\_full(StackType \*s) {  
 return (s->top == (MAX\_STACK\_SIZE – 1));  
}  
// 삽입함수  
void push(StackType \*s, element item) {  
 if (is\_full(s)) {  
 fprintf(stderr, “스택 포화 에러\n”);  
 return;  
 }  
 else s->data[++(s->top)] = item;  
}  
// 삭제함수  
element pop(StackType \*s) {  
 if (is\_empty(s)) {  
 fprintf(stderr, “스택 공백 에러\n”);  
 exit(1);  
 }  
 else return s->data[(s->top)--];  
}  
// 피크함수  
element peek(StackType \*s) {  
 if (is\_empty(s)) {  
 fprintf(stderr, “스택 공백 에러\n”);  
 exit(1);  
 }  
 else return s->data[s->top];  
}  
  
int main(void) {  
 StackType s;  
 init\_stack(&s);  
 int len, half;  
 bool flag = true;  
  
 element string[MAX\_STACK\_SIZE];   
 printf(“문자열을 입력하시오: “);  
 scanf\_s(“%s”, string, sizeof(string));  
 len = strlen(string);  
  
 element alnum[MAX\_STACK\_SIZE];  
 int n = 0;  
 for (int i = 0; i < len; i++) {  
 if (isalnum(string[i])) {  
 alnum[n++] = tolower(string[i]);  
 }  
 }  
  
 half = n / 2;  
 for (int i = 0; i < half; i++) {  
 push(&s, alnum[i]);  
 }  
  
 if (n % 2 != 0) {  
 half++;  
 }  
 for (int i = half; i < n; i++) {  
 if (alnum[i] != pop(&s)) {  
 printf(“회문이 아닙니다.”);  
 flag = false;  
 break;  
 }  
 }  
 if (flag) printf(“회문입니다.”);  
 return 0;  
}  
(C에서 입력값에다 공백을 포함하면 scanf\_s가 공백을 입력값의 끝으로 인식해서 단어 하나만 됨)