# 자료구조와 실습
# 5장 스택 연습문제

학번 : 2016110056
학과 : 불교학부
이름 : 박승원
날짜 : 2016년 10월 17일

1. 스택을 가장 효과적으로 이용하는 기법은?

   (a) 루핑(looping)

   (b) 알고리즘(algorithm)

   (c) 반복(iteration)

   (d) 순환(recursion)

2. 스택에서 삽입 작업이 발생하면 top의 값은?

   (a) top == 0

   (b) top == 1

   (c) top = top - 1

   (d) top = top + 1

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

   (a) A,B,C,D,E

   (b) E,D,C,B,A

   (c) A,B,C,E,D

   (d) B,A,C,D,E

4. 10,20,30,40,50을 스택에 넣었다가 3개의 항목을 삭제하였다. 남아있는 항목은?

   10, 20

5. 다음 중 스택에 대한 설명 중 맞는 것은?

   (a) 스택은 FIFO(First-In First-Out)방식으로 동작한다.

   (b) 스택은 양쪽 끝을 사용하여 입출력을 한다.

   (c) 스택의 삭제 연산보다 스택의 삽입 연산이 훨씬 쉽다.

   (d) 스택은 중간에서 요소를 삭제하는 것을 허용하지 않는다.

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

   (a) 1

   (b) 2

   (c) 3

   (d) 4

7. 다음 중 배열로 구현된 스택에서 공백 상태에 해당하는 조건은? 또 포화 상태에 해당되는 조건은?

   (a) top == -1

   (b) top == 0

   (c) top == (MAX_STACK_SIZE-1)

   (d) top == MAX_STACK_SIZE

   정확히는 스택의 구현에 따라 달라진다.

8. 다음 중 연결 리스트로 구현된 스택에서 공백 상태에 해당하는 조건은?

   (a) top == NULL
   (b) *top == NULL
   (c) *top == MAX_STACK_SIZE-1
   (d) top == MAX_STACK_SIZE

9. 스택에 항목들을 삽입하고 삭제하는 연산은 시간 복잡도가 어떻게 되는가?

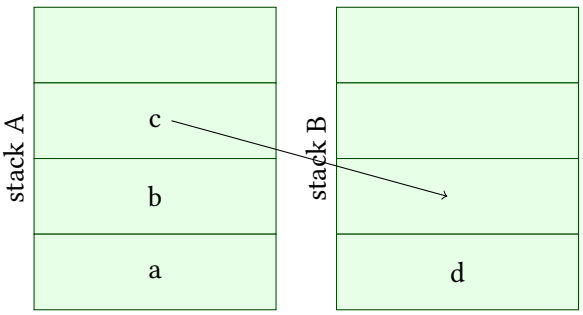
   (a) O(1)
   (b) O($log_2 n$)
   (c) O(n)
   (d) O($n^2$)

10. 다음 중 스택이 사용될 수 있는 상황은?

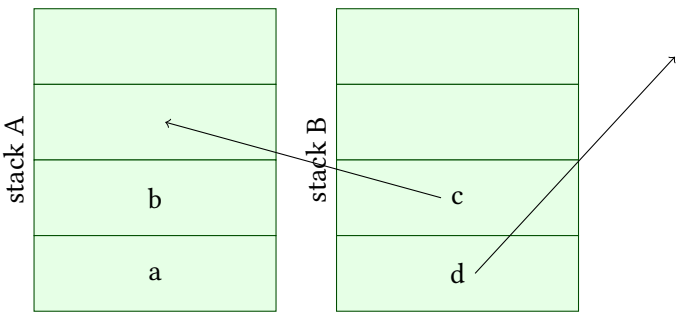
    (a) UNDO 기능을 구현하기 위하여 실행된 명령어들을 기억할 때
    (b) 키보드에서 입력된 키스트로크를 잠시 저장할 때
    (c) 다항식의 항들을 저장할 때
    (d) 회사에서 입사순으로 승진시킬 때

11. 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);



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

| | | |
|---|---|---|
| push(A,1); | 1 | 1 \| | | | |
| push(A,2); | 2 | 1 \| 2 \| | | | |
| push(A,3); | 3 | 1 \| 2 \| 3 \| | | |
| pop(A); | 2 | 1 \| 2 \| | | | |
| push(A,4); | 3 | 1 \| 2 \| 4 \| | | |
| push(A,5); | 4 | 1 \| 2 \| 4 \| 5 \| | |
| push(A,6); | 5 | 1 \| 2 \| 4 \| 5 \| 6 \| |
| push(A,7); | 5 | 1 \| 2 \| 4 \| 5 \| 6 \| |
| pop(A); | 4 | 1 \| 2 \| 4 \| 5 \| | |

13. 연결된 스택 A에 다음과 같이 삽입과 삭제가 되풀이되었을 경우에 각 단계에서의 연결된 스택의 내용(노드, top이 가리키는 값)을 나타내시오.

| | |
|---|---|
| push(A,1); | 1 |
| push(A,2); | 2 |
| push(A,3); | 3 |
| pop(A); | 2 |
| push(A,5); | 5 |
| push(A,6); | 6 |
| pop(A); | 5 |

14. 만약 연결된 스택의 C언어의 구현에서 만약 저장하려는 항목이 정수가 아니고 다음과 같은 구조체라면 소스의 어떤 부분들이 변경되어야 하는가?

```
typedef struct {
    char name[MAX_NAME_SIZE];
    int student_no;
} element;
```

데이터 element에 맞게 삽입, 입력, 출력 부분을 바꾸어준다.

15. 괄호 검사 프로그램에서 다음의 입력을 처리한다고 가정할 때, 스택에 최대로 쌓이게 되는 괄호의 개수는 몇 개인가?

(([]{{{}}}))

5개

16. 알고리즘 5.1의 괄호 검사 프로그램에서 다음과 같은 수식이 주어졌을 경우, 알고리즘을 추적하여 각 단계에서의 스택의 내용을 그려라.

(a) ab[(c+d)*e]-f

(b) (a(b*c)/[d+e]/f)-g

17. 다음은 어떤 수식의 후위 표기이다. 이 때 최초로 수행되는 연산은 어느 것인가?

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

(a) B+E

(b) E+A

(c) D*B

(d) B*E

18. 후위 표기식 계산 프로그램에서 다음과 같은 수식이 주어졌을 경우, 알고리즘을 추적하여 각 단계에서의 스택의 내용을 그려라. a=1, b=2, c=3, d=4, e=4라고 가정하라.

(a) ab*ca-/de*+

(b) ab-c*d+

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

```
int size(Stack* st) {
   return st->top;
}
```

20. 연결 리스트로 구현된 스택에 저장된 요소의 수를 반환하는 size 연산을 구현하여 보라.

```
int size(Stack* st) {
   if(!st) return 0;
   return 1 + size(st->node);
}
```

21. 배열을 이용한 스택의 문제점은 스택을 생성할 때 MAX_STACK_SIZE를 결정해야 한다는 것이다. 이 결점을 보완하는 한 가지 방법은 max_top이라는 변수를 도입하여 스택이 만들어질 때는 max_top을 0으로 하여 시작하고 만약 삽입 연산 중에 새로운 요소를 추가할 공간이 없을 때에는, max_top을 max_top*2+1으로 변경하고 변경된 크기만큼의 공간을 동적으로 할당하여 새로운 배열을 만든다. 요소들은 이전 배열에서 새로운 배열로 복사되고 이전 배열은 지워진다. 비슷하게 만약 삭제 연산 중에 요소들의 개수가 배열 크기의 1/4로 떨어지게 되면 기존 크기의 절반인 배열을 생성하고 이전 요소들을 복사한 다음, 이전의 배열을 삭제한다. 이 스택을 구현하고 테스트하라.

```
#include<stdio.h>
#include<stdlib.h>
typedef int element;
typedef struct {
element *stack;
   int top;
   int max_top; //현재 배열의 크기
} RStackType;

void change_stack_size(RStackType* st, int sz) {
   element* rs = (element*)calloc(sz, sizeof(element));
   for(int i=0; i<st->top; i++) rs[i] = st->stack[i];
   if(st->stack) free(st->stack);
   st->stack = rs;
```

```
}

void push(RStackType* st, element n) {
   if(st->top == st->max_top) {
      st->max_top = 2 * st->max_top + 1;
      change_stack_size(st, st->max_top);
   }
   st->stack[st->top++] = n;
}

void pop(RStackType* st) {
   if(st->top < st->max_top / 4) {
      st->max_top = st->max_top / 2;
      change_stack_size(st, st->max_top);
   }
   st->top--;
}

void show(RStackType* st) {
   for(int i=0; i<st->top; i++) printf("%d ", st->stack[i]);
   printf("\n");
}

int main() {
   RStackType rs = {NULL, 0,0};
   for(int i=0; i<100; i++) push(&rs, i);
   show(&rs);
   for(int i=0; i<90; i++) pop(&rs);
   show(&rs);
}
```

```
zezeon@ubuntuZ:~/Programming/exercise$ rm 21.png
zezeon@ubuntuZ:~/Programming/exercise$ make 21.png
---------- 문제 21번 실행을 시작합니다. --------------------------
./21.x
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58
 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 8
6 87 88 89 90 91 92 93 94 95 96 97 98 99
0 1 2 3 4 5 6 7 8 9
---------- 문제 21번 실행을 종료합니다. ----------------------------
```

22. 본문에는 단순 연결 리스트로 구현된 스택을 소개하였다. 4장 리스트에서 배운 이중 연결 리스트를 사용하여 연결된 스택을 구현하여 보라.

소감 이것도 미리 해둔 것이라 같이 제출합니다. 그런데, 스택을 이중 연결 리스트로 구현하는 것은 아무 의미가 없는 일이 아닌가 의심이 든다.