자료구조응용
07. 스택과 큐

1. 후위표기법(postfix notation)으로 표현된 하나의 수식을 파일(input.txt)로 입력받아 그 계산결과를 화면에 출력하는 프로그램을 작성하라.

[프로그램 설명]

입력파일(input.txt) : 82/3-42*+
사용되는 연산자 : +, -, *, /, %
사용되는 피연산자 : 1~9 사이의 한 자리 정수
'(', ')' 연산자는 입력되지 않음
divide by zero에 대한 테스트 및 처리는 구현하지 않음
입력수식의 문자열 길이는 최대 80으로 함

```
int stack[MAX_STACK_SIZE];
int top = -1;
```

```
typedef enum {lparen, rparen, plus, minus, times, divide,
                        mod, eos, operand} precedence;
```

```
int eval(void)
{/* evaluate a postfix expression, expr, maintained as a
    global variable. '\0' is the the end of the expression.
    The stack and top of the stack are global variables.
    getToken is used to return the token type and
    the character symbol. Operands are assumed to be single
    character digits */
  precedence token;
  char symbol;
  int op1, op2;
  int n = 0; /* counter for the expression string */
  top = -1;
  token = getToken(&symbol, &n);
  while (token != eos) {
     if (token == operand)
        push(symbol-'0'); /* stack insert */
     else {
        /* pop two operands, perform operation, and
           push result to the stack */
        op2 = pop(); /* stack delete */
        op1 = pop();
        switch(token) {
           case plus: push(op1+op2);
                      break;
           case minus: push(op1-op2);
                       break;
           case times: push(op1*op2);
                       break;
           case divide: push(op1/op2);
                        break;
           case mod: push(op1%op2);
        }
     }
     token = getToken(&symbol, &n);
  }
  return pop(); /* return result */
}
```

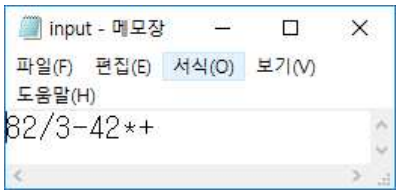
Program 3.13: Function to evaluate a postfix expression

```
precedence getToken(char *symbol, int *n)
{/* get the next token, symbol is the character
    representation, which is returned, the token is
    represented by its enumerated value, which
    is returned in the function name */
  *symbol = expr[(*n)++];
  switch (*symbol) {
     case '(' : return lparen;
     case ')' : return rparen;
     case '+' : return plus;
     case '-' : return minus;
     case '/' : return divide;
     case '*' : return times;
     case '%' : return mod;
     case '\0' : return eos;
     default  : return operand; /* no error checking,
                                   default is operand */
  }
}
```

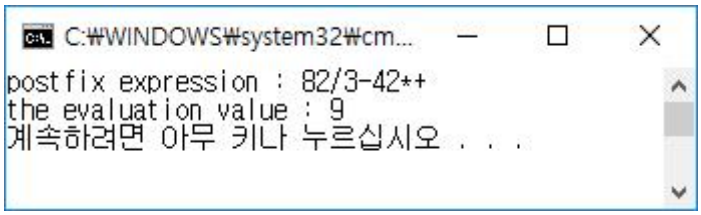
Program 3.14: Function to get a token from the input string

[실행 예]

입력 파일



화면 출력



2. 중위표기법(infix notation)으로 표현된 하나의 수식을 파일로 입력받아 후위표기법(postfix notation)으로 변환하여 화면 및 파일에 동시에 출력하는 프로그램을 작성하라.

[프로그램 설명]

입력파일("input.txt") : (4/(2-2+3))*(3-4)*2 화면출력 & 파일출력("output.txt") : 422-3+/34-*2* ※ 입력수식의 문자열 길이는 최대 80으로 함
사용되는 연산자 : +, -, *, /, %, (,) 사용되는 피연산자 : 알파벳 소문자, 1~9 사이의 한 자리 정수 ※ 피연산자가 모두 1~9의 한 자리 정수면, 출력결과를 2번 문제의 입력으로 사용 가능
posfix(), printToken() 함수의 화면출력 부분에 파일출력 추가 void printToken(precedence); 직접 구현

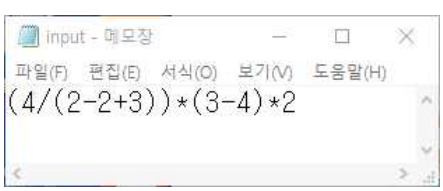
```
typedef enum {lparen, rparen, plus, minus, times, divide,
                          mod, eos, operand} precedence;

/* isp and icp arrays -- index is value of precedence
   lparen, rparen, plus, minus, times, divide, mod, eos */
int isp[] = {0,19,12,12,13,13,13,0};
int icp[] = {20,19,12,12,13,13,13,0};

precedence stack[MAX_STACK_SIZE];

top = -1;
```

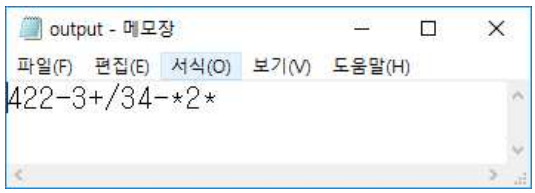
```
void postfix(void)
{/* output the postfix of the expression. The expression
    string, the stack, and top are global */
  char symbol;
  precedence token;
  int n = 0;
  top = 0;    /* place eos on stack */
  stack[0] = eos;
  for (token = getToken(&symbol, &n); token != eos;
                          token = getToken(&symbol,&n)) {
    if (token == operand)
      printf("%c", symbol);
    else if (token == rparen) {
      /* unstack tokens until left parenthesis */
      while (stack[top] != lparen)
        printToken(pop());
      pop();  /* discard the left parenthesis */
    }
    else {
      /* remove and print symbols whose isp is greater
         than or equal to the current token's icp */
      while(isp[stack[top]] >= icp[token])
        printToken(pop());
      push(token);
    }
  }
  while ( (token = pop()) != eos)
    printToken(token);
  printf("\n");
}
```

Program 3.15: Function to convert from infix to postfix

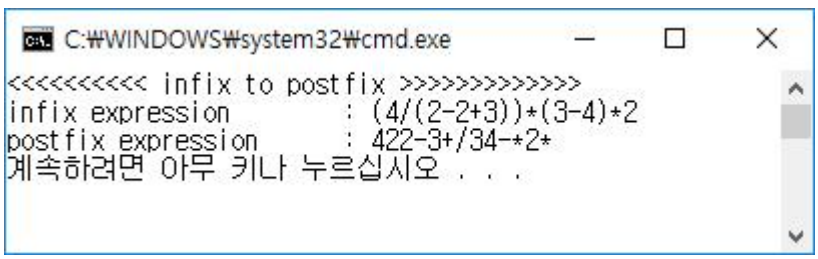
[실행 예]

입력과 출력 파일





화면 출력



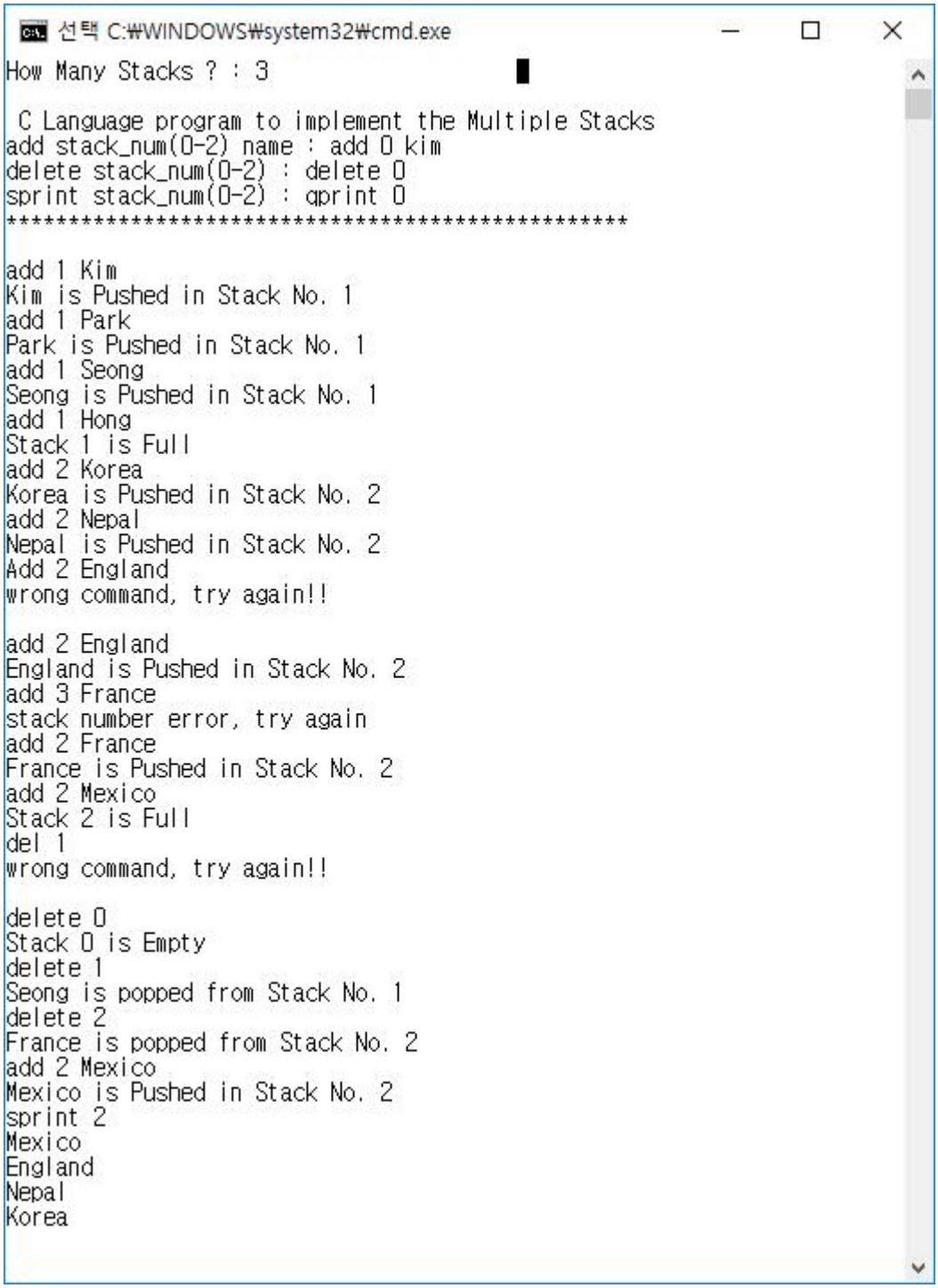
3. 한 개의 arry를 사용하여 multistack을 만들어서 처리하는 프로그램을 작성하시오. 배열의 크기는 10이며 배열안의 원소는 스트링으로 처리한다. 스택의 개수는 입력을 받아 정한다. 각 스택의 크기는 10/(스택수)로 균일하게 한다. 스택의 번호는 0번부터 시작한다. 스택의 수가 3개아면 스택번호는 0, 1, 2 가 된다. 스택의 명령어는 다음과 같다.

add 스택번호 스트링 : 해당스택에 스트링 삽입
delete 스택번호 : 해당 스택에 스트링 삭제
sprint 스택번호 : 해당스택안의 top부터 bottom 내용을 출력
명령어 오류시 “wrong command, try again!!” 메시지 출력
스택번호 오류시 “stack number error, try again” 메시지 출력
실행의 예를 참조하여 프로그램을 작성할 것.

```
#define max_s 10
#define name_s 10
typedef struct {
        char name[name_s];
} s_data;
s_data stack[max_s];
```

[실행 예]

화면 출력


선택 C:₩WINDOWS₩system32₩cmd.exe
How Many Stacks ? : 3
C Language program to implement the Multiple Stacks
add stack_num(0-2) name : add 0 kim
delete stack_num(0-2) : delete 0
sprint stack_num(0-2) : qprint 0
****
add 1 Kim
Kim is Pushed in Stack No. 1
add 1 Park
Park is Pushed in Stack No. 1
add 1 Seong
Seong is Pushed in Stack No. 1
add 1 Hong
Stack 1 is Full
add 2 Korea
Korea is Pushed in Stack No. 2
add 2 Nepal
Nepal is Pushed in Stack No. 2
Add 2 England
wrong command, try again!!
add 2 England
England is Pushed in Stack No. 2
add 3 France
stack number error, try again
add 2 France
France is Pushed in Stack No. 2
add 2 Mexico
Stack 2 is Full
del 1
wrong command, try again!!
delete 0
Stack 0 is Empty
delete 1
Seong is popped from Stack No. 1
delete 2
France is popped from Stack No. 2
add 2 Mexico
Mexico is Pushed in Stack No. 2
sprint 2
Mexico
England
Nepal
Korea

■ 제출 형식

- 공학인증 시스템(ABEEK)에 과제를 올릴 때 제목:
- 1차 제출: 학번_이름_DS_07(1), 2차 제출: 학번_이름_DS_07(2)
- 솔루션 이름 : DS_07
- 프로젝트 이름 : 1, 2, 3
- 실행화면을 캡쳐하여 한글파일에 추가 후 솔루션 폴더에 포함.
- 한글 파일명 : 학번_이름.hwp
- 솔루션 폴더를 압축하여 제출할 것.
- 솔루션 압축 파일 명:

  1차 제출: 학번_이름_DS_07(1).zip, 2차 제출: 학번_이름_DS_07(2).zip

- 제출은 2회 걸쳐 가능(수정 시간 기준으로 처리)