1)Main Heading

```
*********************************************************************************************
* Name : 차윤성
* Student ID : 20163162
* Program ID : Hw#2
* Description : 중위 표기되어있는 수식을 후위 표기로 바꾼 후에 후위 연산자 연산 알고리즘을 통해 결과를
* 도출하는 프로그램
* Algorithm :   먼저 중위 표기로 되어있는 수식을 차례대로 검사하며 숫자 부분은 그대로 출력 한 후 사칙연
*               산자가 나오면 연산자를 스택에 저장한다. 그 후 숫자를 또 출력한 후에 새로 연산자가 나타나
*               면 기존 스택에 있는 연산자와 새로운 연산자의 우선순위를 검사 한 후에 기존 스택에 있는 연
*               산자의 우선순위가 높으면 스택에 있는 연산자를 꺼내서 출력한 후에 새로운 연산자를 스택에
*               넣고 그렇지 않은 경우에는 새로운 연산자를 스택에 바로 넣는다. 나중에 수식이 끝나면 스택
*               에 남아있는 모든 연산자를 꺼내어 출력하여 후위 표기로 바꾼다.
*               후위 표기로 바꾼 수식은 후위 연산자 연산 알고리즘을 통해 결과를 도출하는데, 이 알고리즘
*               은 반대로 숫자를 스택에 넣고 연산자가 등장하면 스택에 들어있는 2개의 숫자를 꺼내와 연산
*               자에 맞는 연산을 한 후에 다시 스택에 넣는다. 이 방식을 반복 실행하여 연산의 결과를 도출
*               한다.
* Variables :   class ArrStack{} : ArrStack class의 내용을 정의한다.
*               bool isFull() : class ArrStack의 stack이 가득 찼는지 검사한다.
*               bool isEmpty() : class ArrStack의 stack이 비어있는지 검사한다.
*               void push(char data) : class ArrStack의 stack에 data를 넣는다.
*               char pop() : class ArrStack의 stack에 들어있는 값을 하나 제거하고 출력한다
*               int priority(char input) : input 연산자의 연산 우선순위를 반환해주는 함수이다.
*               void infixToPostfix(char *buffer) : buffer배열, 즉 입력받은 수식을 후위연산으로 변환한다.
*               int postfixEvaluation(char *buffer) : buffer배열, 즉 입력받은 후위 표기 수식을 연산하는 함수이다.
*               char buffer[MAX_SIZE] : 파일 입출력을 통해 입력받은 수식을 저장하는 배열 변수이다.
*               char bufEval[MAX_SIZE] : 후위 연산으로 바꾼 수식을 저장하는 배열변수이다.
*********************************************************************************************
#include <iostream>
#include <fstream>
#define MAX_SIZE 100 // 최대 크기를 100으로 설정

char bufEval[MAX_SIZE]; // 후위 연산으로 바꾼 수식을 저장하는 배열
// 따로 선정해 준 이유는 후위연산 알고리즘을 통해 결과를 뽑기 위해서
using namespace std;
```

2-1) Sub Heading

```
*********************************************************************************************
* class : ArrStack
* description : ArrStack의 사용 목적은 stack을 만들고 stack의 size와 top 포인터를 지정하기 위함이다.
* variables :   int top : stack의 현재 값을 반환하기 위한 인덱스 변수이다. 현재 stack에 들어있는 값이 없
*                         으므로 -1로 초기화해주었다.
*               char *stack : stack을 stack이라는 배열로 만들었다.
*               int max_size : stack의 최대 크기를 지정하기 위한 변수이다.
*               bool isFull() : top이 max_size와 같다면 stack이 가득 찼으므로 true, 아니면 false를 반환한다.
*               bool isEmpty() : top이 -1이라면 stack이 비었으므로 true,아니라면 false를 반환한다.
*               void push(int data) : stack이 가득 찼는지 검사 후에 data를 top이 가리키는 위치에 넣어준다.
*               char pop() : stack이 비어있는지 검사 후에 top이 가리키는 위치에 있는 값을 꺼내어 출력
*                            한 후에 제거한다.
*********************************************************************************************

class ArrStack {
public:
        int top;
        char *stack;
        int max_size;

        ArrStack(int size) {
                top = -1;
                stack = new char(size);
                max_size = size;
        }

        bool isFull();
        bool isEmpty();
        void push(char data);
        char pop();
};
```

2-2) Sub Heading

```
*********************************************************************************************
* function : isFull
* description : isFull의 목적은 stack이 가득 차있는지 아닌지 검사하는 것이다.
*********************************************************************************************
bool ArrStack::isFull() {
	if (top + 1 == max_size) return true; // top+1이 max_size와 같으면 가득 찼다는 의미
	else return false;
}
```

2-3) Sub Heading

```
*********************************************************************************************
* function : isEmpty
* description : isEmpty의 목적은 stack이 비어있는지 아닌지 검사하는 것이다.
*********************************************************************************************
bool ArrStack::isEmpty() {
	if (top == -1) return true; // top이 -1이면 stack이 비어있다는 의미
	else return false;
}
```

2-4) Sub Heading

```
*********************************************************************************************
* function : push
* description : push의 목적은 stack에 값을 넣어주는 것이다.
* variables : char data : 함수의 매개변수. 매개변수로 들어온 값을 stack에 넣어준다.
*********************************************************************************************
void ArrStack::push(char data) {
	if (isFull()) cout << "Stack is Full" << endl;
	else {
		top++;   // top의 크기를 1 늘려준 후
		stack[top] = data;// 그다음 인덱스에 data 삽입
	}
}
```

2-5) Sub Heading

```
*********************************************************************************************
* function : pop
* description : pop의 목적은 기존 stack에 들어있는 1개의 값을 반환한 후 값을 제거하는 것이다.
*********************************************************************************************
char ArrStack::pop() {
	if (isEmpty()) {    // stack이 비어있는지 검사
		cout << "Stack is empty" << endl;
		return 0;
	}
	else return stack[top--]; // stack[top]에 있는 걸 반환 후 top의 크기를 1 줄임
}
```

2-6) Sub Heading

```
*********************************************************************************************
* class : IntStack
* description : IntStack의 목적은 ArrStack과 기능은 같지만 후위 표기 수식의 연산 시 기능하기 위해 Int형
*               으로 만들었다.
* variables :   int top : stack의 현재 값을 반환하기 위한 인덱스 변수이다. 현재 stack에 들어있는 값이 없
*                         으므로 -1로 초기화해주었다.
*               int *stack : stack을 stack이라는 배열로 만들었다.
*               int max_size : stack의 최대 크기를 지정하기 위한 변수이다.
*               bool isFull() : top이 max_size와 같다면 stack이 가득 찼으므로 true, 아니면 false를 반환한다.
*               bool isEmpty() : top이 -1이라면 stack이 비었으므로 true,아니라면 false를 반환한다.
*               void push(int data) : stack이 가득 찼는지 검사 후에 data를 top이 가리키는 위치에 넣어준다.
*               int pop() : stack이 비어있는지 검사 후에 top이 가리키는 위치에 있는 값을 꺼내어 출력
*                           한 후에 제거한다.
*********************************************************************************************
class IntStack {
public:
	int top;
	int* stack;
	int max_size;

	IntStack(int size) {
		top = -1;
		stack = new int[size];
		max_size = size;
	}
```

```
        bool isFull();
        bool isEmpty();
        void push(int data);
        int pop();
};
bool IntStack::isFull() {
        if (top + 1 == max_size) return true; // top+1이 max_size와 같으면 가득 찼다는 의미
        else return false;
}

bool IntStack::isEmpty() {
        if (top == -1) return true; // top이 -1이면 stack이 비어있다는 의미
        else return false;
}
```

2-7) Sub Heading

```
*********************************************************************************************
* function : push
* description : push의 목적은 stack에 값을 넣어주는 것이다.
* variables : int data : 함수의 매개변수. 매개변수로 들어온 값을 stack에 넣어준다.
*********************************************************************************************
void IntStack::push(int data) {
        if (isFull()) cout << "Stack is Full" << endl;
        else {
                top++;   // top의 크기를 1 늘려준 후
                stack[top] = data;// 그다음 인덱스에 data 삽입
        }
}

int IntStack::pop() {
        if (isEmpty()) {    // stack이 비어있는지 검사
                cout << "Stack is empty" << endl;
                return 0;
        }
        else return stack[top--]; // stack[top]에 있는 걸 반환 후 top의 크기를 1 줄임
}

int priority(char input);
void infixToPostfix(char *buffer);
int postfixEvaluation(char *buffer);

int main() {
        char buffer[MAX_SIZE];

        ifstream infile("hw2.txt");

        while (infile.getline(buffer, MAX_SIZE)) {      // 데이터 파일을 읽어옴
                cout << "1) Echo data\t(infix form)\t: " << buffer << endl;
                cout << "2) Conversion\t(postfix form)\t: ";
                infixToPostfix(buffer);
                cout << endl;
                cout << "3) Result\t\t\t: " << postfixEvaluation(bufEval) << endl;
        }
}
```

2-8) Sub Heading

```
*********************************************************************************************
* function : priority
* description : priority의 목적은 각각의 연산자 별로 우선순위를 설정해주는 것이다.
* variables : char input - 함수의 매개변수, 들어온 값의 우선순위를 설정해준다
*                                 단, -1의 경우는 데이터 파일의 공백을 무시하기 위해 설정하였다.
*********************************************************************************************
int priority(char input) {
        // 각각의 return값은 각 연산자 별 연산 우선순위를 의미
        if (input == ')') return 4;
        else if (input == '*' || input == '/') return 3;
        else if (input == '+' || input == '-') return 2;
        else if (input == '(') return 1;
        else if (input == ' ') return -1; // 데이터 파일에 공백이 있어서 -1을 반환하게 한 후 무시하도록 함
        else return 0;

}
```

2-9) Sub Heading

```
*********************************************************************************************
* function : infixToPostfix
* description : infixToPostfix의 목적은 중위 표기로 되어있는 수식을 후위 표기로 바꿔주는 것이다.
* variables :    int i - 매개변수로 들어오는 buffer, 즉 중위 표기 수식이 저장된 배열의 인덱스 변수
*                int j - 후위 표기로 바꾼 수식의 배열이 저장되는 bufEval배열에 수식을 저장해주기위한 인덱
*                        스 변수
*                stkCon - ArrStack의 객체 인스턴스, Conversion에 쓰일 stack을 생성한다.
*********************************************************************************************
void infixToPostfix(char *buffer) {
        ArrStack stkCon(MAX_SIZE)
        int i = 0, j = 0;
        while (buffer[i] != '\0') {
                switch (priority(buffer[i])) { // 연산 우선순위 별로 실행되도록 함
                case 0: // 숫자일 경우는 stack 에 push
                        bufEval[j++] = buffer[i]; // 숫자를 bufEval에 저장
                        cout << buffer[i] << " ";
                        break;
                case 4: // ')'일 경우 stack의 top에 '('가 나올때까지 pop
                        while (stkCon.stack[stkCon.top] != '(') {
                                bufEval[j] = stkCon.pop();
                                cout << bufEval[j++] << " ";
                        }
                        break;
                case 3: // 연산자가 *, /일 경우 + -와 처리 방법은 같으므로 case 2 와 똑같이 처리되도록 함
                case 2:
                        if (priority(buffer[i]) >= priority(stkCon.stack[stkCon.top])) stkCon.push(buffer[i]);
                        else{// 새로 들어온 연산자가 기존 스택에 있는 연산자보다 우선순위가 높으면 push.
                                bufEval[j] = stkCon.pop();  // 우선순위가 낮으면 pop
                                cout << bufEval[j++] << " ";
                                stkCon.push(buffer[i]);
                        }
                        break;
                case 1: // '('경우 바로 stack에 push - 우선순위가 제일 낮으므로
                        stkCon.push(buffer[i]);
                        break;

                default: // buffer의 내용이 공백일경우 무시하도록 함
                        break;
                }
                i++;
        }

        while (stkCon.top != -1) { // 모든 연산자를 검사한 후에 스택에 남아있는 모든 걸 출력
                if (stkCon.stack[stkCon.top] == '(') { // '('은 출력이 안되도록 함
                        stkCon.pop();
                        continue;
                }
                else {
                        bufEval[j] = stkCon.pop();
                        cout << bufEval[j++] << " ";
                }
        }
        bufEval[j] = '\0'; // 마지막에 Null문자 적용
}
```

2-10) Sub Heading

```
*************************************************************************************************
* function : postfixEvaluation
* description : postfixEvaluation의 목적은 후위 표기 수식을 연산하는 것이다.
* variables :    int i - 매개변수로 들어오는 buffer, 즉 후위 표기 수식을 저장한 배열의 인덱스 변수
*                int op1,op2 - 연산자가 등장할 경우 stack에서 pop한 두 개의 정수를 저장해주는 변수
*
*************************************************************************************************
int postfixEvaluation(char *buffer) {
        IntStack stkEval(MAX_SIZE);

        int i = 0;
        int op1,op2;
        while (buffer[i] != '\0') { // 똑같이 연산자의 우선순위를 검사하지만 '(',')'가 없으므로 사칙연산만 검사
                switch (priority(buffer[i])) {
                case 0:  // 연산자가 아닌 숫자일 때 stack에 push
                        stkEval.push(buffer[i]-'0'); // char형 정수를 int형 정수로 변환하는 방법 : char형 정수 - '0'
                        break;
                case 3:
                case 2:
                        op2 = stkEval.pop();
                        op1 = stkEval.pop();

                        switch (buffer[i]) { // 각 연산자 별로 연산한 값을 다시 stack에 push
                        case '+':
                                stkEval.push(op1 + op2);
                                break;
                        case '-':
                                stkEval.push(op1 - op2);
                                break;
                        case '*':
                                stkEval.push(op1 * op2);
                                break;
                        case '/':
                                stkEval.push(op1 / op2);
                                break;
                        }
                        break;
                default: // '(',')' 또는 공백일 경우 무시
                        break;
                }
                i++;
        } // 마지막까지 stack에 남아있는 값은 최종 연산 값
        return stkEval.pop();
}
```