2017년도 2학기 컴퓨터 프로그램 설계 과제 – HW5-

1. 급여 관리 프로그램

입력 파일: input.txt은 다음과 같은 형태로 구성되어 있다. 첫 번째 행은 총 데이터의 수이고 이름, 전화번호, 근무시간, 시간당 임금의 순으로 저장되어 있다.

```
3
박남일
010-1111-1111
20
20000
이범종
010-2222-2222
20
15000
이수연
010-3333-3333
20
30000
```

출력 파일(output.txt)에는 임금(근무시간*시간당 임금)을 기준으로 내림차순 정렬하여 파일에 출력(저장)하도록 작성한다. 예를 들어 위의 입력파일에 대한 출력 파일(output.txt)는 다음과 같다.

Name	Phone	Working Hour	pay/hour	total pay
이수연	010-3333-3333	20	30000	600000
박남일	010-1111-1111	20	20000	400000
이범종	010-2222-2222	20	15000	300000

(힌트) person의 구조체는 다음과 같이 선언할 수 있다.

```
struct person {
        char name[50];
        char phone[20];
        int pay_hour;
        int work_hour;
        int total_pay;
};
```

2. 문자일로 입력된 정수를 한글로 출력하기

최대 20자리까지의 10진수를 문자열로 읽어서 한글로 출력하는 프로그램을 작성해 보자. 예를 들어서 "34567"을 입력하면 "삼만 사천오백육십칠'로 출력한다. 출력을 보기 쉽게 하도록 만, 고, 경 등의 단위마다 한 칸씩 뛰어쓰기를 한다.

(힌트) char han[10][4] = {"", "일", "이", "삼", "사", "오", "육", "칠", "팔", "구"};
char smallunit[][4] = {"", "십", "백", "천"};
char largeunit[][4] = {"", "만", "억", "조", "경"};
입력된 문자열의 길의를 4로 나눈 몫은 최상위 자리의 큰 단위를 결정하고 4로 나눈 나머지가 최상위 자리의 작은 단위를 결정한다.

3. 실수 저장

컴퓨터에서 실수를 저장하는 방법은 실수를 다음의 폼으로 변환하여 significand와 exponent를 저장하는 방법이 있다.

$$\text{significand} \times \text{base}^{\text{exponent}}$$

예를 들어, 12.345 = 1.2345×10^1 로 표현할 수 있다. 본 프로젝트에서는 Key board로 문자열을 입력 받아 먼저 규칙에 맞게 표현된 실수인지를 판단한 후 해당 입력에 대한 significand 값과 exponent 값을 구하는 프로그램을 작성한다. 실제 컴퓨터에서는 이진수만 처리하므로 base가 2로 하여 2진수로 처리하지만 본 프로젝트에서는 우리가 익숙한 base는 10으로 가정한다. 그리고 significand는 항상 소수점 왼쪽이 유효숫자 한자리로 표현한다고 가정한다. 이렇게 약속하면 컴퓨터에는 significand와 exponent만 저장하면 실제값을 계산할 수 있다. 즉, 1234.5는 significand가 1.2345가 되고 exponent는 3이 된다.

사용자가 입력할 '1.2345'와 같은 소수점이 있는 숫자 표기법의 규칙은 다음과 같다.

1) 숫자, 소수점(period), 부호(+,-)로만 이루어진다. 영문자나 특수문자는 사용될 수 없다.
2) 부호는 숫자가 나타나기 전에만 선택적으로 나타날 수 있다. 즉 부호는 생략 가능하며 숫자 중간에서는 사용될 수 없다. 예를 들면, 12-23는 규칙에 어긋남을 의미한다.
3) 0개 혹은 복수개의 숫자 문자 뒤에 소수점이 나타날 수 있다. 소수점 이후에도 0개 혹은 복수 개의 숫자가 나타날 수 있다. 소수점은 오직 한번만 사용 가능하다. 예를 들면, '12.34.56' 등은 올바른 숫자 표현법이 아니다.

Key board로 입력된 문자열이 위와 같은 규칙에 맞게 표현된 숫자인지를 판단한 후 만약 규칙에 맞으면 significand 값과 exponent 값을 출력하고, 틀리면 어떤 부분이 틀렸는지를 출력한다. 예를 들면, 규칙에 맞게 표현된 경우라면 주어진 입력 실수에 대한 significand와 exponent 값을 출력하고, 주어진 문자열이 규칙에 맞지 않는 경우라면 다음과 같이 어떤 오류가 있는지 출력한다. 즉, 영문자가 쓰인 경우에는 "영문자는 숫자의 표기에 사용할 수 없음"이라고 출력하고, 만약 period(.)가 두 번 사용된 경우에는 "소수점은 한 번만 사용될 수 있음"이라고 출력한다. 문자열 '+', 혹은 '-' 이외의 특수 문자로 시작하면 "부호는 + 혹은 – 만이 가능함" 등으로 사용자에게 오류 내용을 친절하게 알리는 메시지를 출력한다. 문자열 중간에 '+' 혹은 '-' 문자가 나타나면 "숫자 중간에 부호는 나타날 수 없음"이라고 출력하고, 기타 특수 문자가 사용되었으면 "특수 문자는 숫자 표현에 사용될 수 없음"이라고 출력한다.

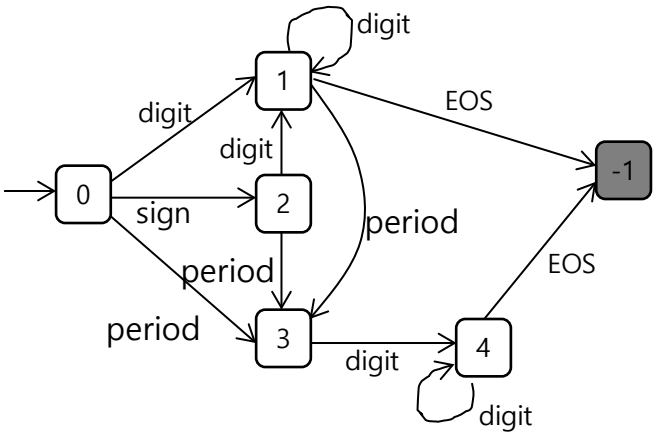
(테스트 케이스)

구현된 프로그램은 아래 표와 같은 테스트 케이스를 통과하여야 한다.

테스트 케이스		테스트 입력	결과
규칙에 맞음		123	significand = 1.23 exponent = 2
		1.234	significand = 1.234 exponent = 0
		+.123	significand = 1.23 exponent = -1
		-.123	significand = -1.23 exponent = -1
규칙 위반	부호 오류	*123, 1-2, 1.+234, 1.23+45	부호 오류
	영문자 사용	1A23, +A123, 12.A, 12.3A	영문자 사용
	특수문자 사용	12#34, +12.$, 12.34*	특수문자 사용
	소수점 사용 오류	12.34.56	소수점 사용 오류
	숫자 없음	. +	숫자 없음

(프로그래밍 힌트)

(1) 규칙을 아래 [그림9.1] 과 같은 finite automaton으로 표현하는 것이 프로그램을 쉽게 만든다



표시

EOS : End of String
digit : {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
alfa : {a, …, z, A,… Z}
sign : { +, - }
period : { . }
special : 그 밖의 특수 문자

[그림1] Number Recognizer

위 automaton를 array로 표현한 후, 입력된 문자열 in[ ]의 각 문자 in[k],k=0,..,strlen(in)-1 에 대하여

- 문자의 종류(위 [그림1]의 표시 참조)를 구별하고,
- 문자의 종류에 따라 다음 상태를 결정한다.
- 상태 천이(transition)가 불가능한 문자이면, 상태와 문자의 유형에 따라 적절한 error message를 출력한다. 예를 들면, 1의 상태에서 sign ('+' 나 '-') 를 만나면 "숫자 중간에 부호는 나타날 수 없음"이라는 메시지를 출력한다.
- 에러가 발견되면 문자열 분석을 중지하고 프로그램을 종료한다.

(2) Finite automaton을 다음과 같은 array로 표현할 수 있다.

빈 칸을 채운 후, 이 표를 프로그램 내에 array 변수 초기 값으로 설정한다. 표에서 -1은 규칙에 맞는 문자열을 인식하였음을 가리키며, -10 이하의 수는 모두 규칙에 어긋나게 표현된 것임을 나타낸다. 예를 들면, -12는 "empty 문자열임"을 의미하고, "-10"은 "부호는 + 혹은 – 만이 가능함" 이라는 메시지를 의미한다.

입력문자 / 상태	sign	digit	period	special	EOS
0	2	1	3	-10	-12
1	-13	1	3	-14	-1
2					
3					
4					