# CSED101. Programming & Problem Solving

# Fall, 2019

## Programming Assignment #5

## (70 Points)

**차동민 (cardongmin@postech.ac.kr)**

■ ***Due:*** **2019.12.14 23:59**

■ ***Development Environment : Windows Visual Studio 2019***

■ **제출물**

- **C Code files (*.c, *.h)**
  - 제출시, 모든 파일을 하나의 파일로 압축해서 제출할 것. (압축파일명 : **assn5.zip**)
  - 프로그램의 소스 코드를 이해하기 쉽도록 반드시 **주석**을 붙일 것.

- **보고서 파일 (.doc(x) or .hwp) 예) assn5.doc(x) 또는 assn5.hwp**
  - AssnReadMe.pdf 를 참조하여 작성할 것.
  - **명예서약(Honor code)** : 표지에 다음의 내용을 포함한다. "나는 이 프로그래밍 과제를 다른 사람의 부적절한 도움 없이 완수하였습니다." 보고서 표지에 명예서약이 없는 경우는 과제를 제출하지 않은 것으로 처리한다.

- 소스코드와 보고서 파일을 LMS를 이용하여 제출한다.

■ **주의사항**

- 각 문제에 해당하는 요구사항을 반드시 지킬 것.
- 모든 문제의 출력 형식은 아래의 예시들과 동일해야 하며, 같지 않을 시는 감점이 된다.
- 문제에 제시되어 있는 파일이름으로 제출 할 것. 그 외의 다른 이름으로 제출하면 감점 또는 0점 처리된다.
- 컴파일 & 실행이 안되면 무조건 0점 처리된다.
- **하루 late시 20%가 감점되며, 3일 이상 지나면 받지 않는다. (0점 처리)**
- 부정행위에 관한 규정은 POSTECH 전자컴퓨터공학부 학부위원회의 'POSTECH 전자컴퓨터공학부 부정행위 정의'를 따른다. (LMS의 과목 공지사항의 제목 [document about cheating]의 첨부파일인 disciplinary.pdf를 참조할 것.)
- 과제 작성시 전역변수는 사용할 수 없으며 사용자 정의 함수를 적절히 작성하도록 한다.
- 이번 과제에서는 추가 기능 구현에 대한 추가 점수는 없습니다.

## ■ Problem 1

**(문제)**

정수를 원소로 가지는 집합(Set)에 대해 기본 기능을 수행하는 프로그램을 작성하자.

**(목적)**

- 동적 할당을 응용한 Structure(구조체)와 Linked List(연결 리스트)의 사용법을 익힌다.
- 다중 소스 파일의 사용법을 익힌다.
- 명령줄인수(argc, argv) 활용 방법을 익힌다.

**(주의사항)**

1. 이번 과제는 총 12개의 명령어(Problem 1: 7개, Problem 2: 5개)를 입력 받아 기능을 수행한다. 명령어 입력은 사용자 입력뿐만 아니라, 파일로도 명령어 입력을 받을 수 있도록 구현한다. (5쪽, 프로그램 실행 예시 참고)
2. 최소한 각 명령어 별로 함수를 정의해서 사용해야 한다.
3. 이번 과제는 여러 개의 파일로 분할해서 작성한다.
   - **set.h, set.c**
     - 집합을 구조체로 정의하고 집합 리스트를 링크드 리스트로 구현한다.
     - 집합의 원소를 구조체로 정의하고 원소 리스트를 링크드 리스트로 구현한다.
     - 집합과 관련된 함수 선언 및 정의를 한다.
   - **assn5.c**
     - main() 함수를 포함하여 필요한 함수들을 정의한다.
     - main() 함수 내에서는 사용자(또는 파일)로부터 명령어를 입력 받아 알맞은 함수를 호출해 처리하는 기능이 구현되어 있어야 한다.
4. 원소 정렬에 대한 함수는 있어도 좋으나, 이번 과제에서는 집합에 원소 하나가 추가 될 때마다 적절한 위치를 찾아 삽입하여 정렬이 되도록 하는 것으로 충분하고 또 권장한다.
5. 이번 과제는 구조체와 연결리스트를 활용하는 것이 목표이므로 문제에 구조체와 연결리스트를 언급한 부분을 배열을 이용하여 해결 시 감점 처리 된다.
6. 명령어의 인자 값은 항상 올바른 형식으로 주어진다고 가정한다.
7. 명시한 에러 처리 외에는 고려할 필요가 없다.

**(설명 및 요구사항)**

➢ 집합은 구조체와 연결리스트를 사용하여 구현하여야 하며, 아래의 구조체를 사용할 수 있다.

```
typedef struct ele
{
        int data;
        struct ele* next;
} Element;
```

```
typedef struct set
{
        char set_name[21];
        int set_size;
        Element* ele_head;
        struct set* next;
} Set;
```

- 하나의 집합을 이루는 원소(Element)들은 연결 리스트로 구현해야 한다.
- 집합은 이름이 중복되지 않는다면 여러 개 선언이 가능하며, 집합(Set)들 역시 연결 리스트로 구현해야 한다.
- 집합이름(set_name)은 최대 20자로 영어와 숫자의 조합으로 구성된다. 단, 공백이 없다고 가정한다. (잘못된 집합이름 입력에 대해서는 고려할 필요가 없다.)

➢ 구현 해야 할 기능은 7개(set, show, is_element, add, pop, clear, quit)로 아래의 명령어 입력 예시 및 설명을 참고하여 해당 기능이 수행되도록 작성한다.

**(1) 초기화면**

- 프로그램을 실행하면 아래와 같이 명령어 입력을 대기하는 ">> "표시가 출력된다. 이 때 사용자는 명령어를 입력 할 수 있다.

```
>>
```

**(2) 집합 생성: *set [집합이름] [원소 수] [원소, ...]***

- 하나의 정수 집합을 생성하기 위한 초기화 작업으로, 'set setA 5 1 2 3 5 4'입력은 setA 이름을 가진 집합을 생성할 때 5개의 원소 (1, 2, 3, 5, 4)를 추가하여 생성하라는 의미이다.
- 이 때, 생성된 집합(set A)은 집합 리스트(연결 리스트)에 생성되는 순으로 저장하여 관리하도록 하며, 생성된 집합(set A)은 입력 받은 정수를 원소로 가지게 되는데, 이 원소들을 연결 리스트에 저장하고 **오름차순으로 정렬**되도록 한다.
- 집합 생성 후, '***show*** *집합이름*' 명령어를 사용하여 해당 집합 원소를 확인할 수 있다.
- 아래 명령어 'set setN 0' 은 원소 수를 0으로 하여 생성한 경우로, 빈 집합이 생성된다.
- "set setA 5 1 2"와 같이 원소 수가 부족한 입력 등 잘못된 명령어 입력 형식은 고려할 필요가 없다. 단, 입력되는 원소에는 중복 원소가 존재할 수 있다.

(실행 예시의 노란색으로 표시된 문자는 사용자 입력에 해당)

```
>> set setA 5 1 2 3 5 4
>> show setA
setA : 1 2 3 4 5
>> set setN 0
>> show setN
setN :
>> set setA 3 7 8 9
>> show setA
setA : 1 2 3 4 5
>>
```

예외 처리)

- 이미 있는 집합이름으로 집합 생성시, 아무 일도 일어나지 않으며 별도의 에러메시지 없이 다시 사용자 입력(>>)을 받을 준비를 한다.
- 위에서 언급했듯이 원소 수가 부족한 입력, (ex: set setA 5 1 2) 원소 수가 틀린 입력, (ex: set setA 3 1 2 3 4) 과 같은 경우는 고려하지 않는다.
- 원소 수가 알맞게 입력되었는데 그 중 중복된 원소가 있는 경우는 존재할 수 있다. (ex: set setA 4 1 1 1 1의 경우, 문제는 없지만 함수 내부에서 중복된 원소는 미리 감지하고 처리하여, 최종적으로 setA에는 한 개의 1만 들어가도록 한다).

**(3) 집합 원소 출력: *show [집합이름]***

- 특정 집합의 원소를 확인하기 위한 명령어로, 위의 예시처럼 'show setA' 명령어를 입력하면 setA라는 이름을 가진 집합의 원소들을 *[집합이름] : [원소1] [원소2] …* 의 형태로 출력한다.
- 공집합의 경우 원소의 출력 없이 집합이름만 출력한다. 위의 예시의 집합 setN은 공집합으로 'show setN' 명령어 입력 시, 집합이름만 출력됨을 볼 수 있다.
- 집합이름 없이 'show'만 입력할 경우, 아래의 예시처럼 현재 존재하는 모든 집합을 원소와 함께 출력한다.

```
>> show
setA : 1 2 3 4 5
setN :
>> show setB
>>
```

예외 처리)

- show, is_element, add, pop, clear 명령어 모두, 존재하지 않는 집합의 이름이 같이 입력된 경우에 아무 일도 일어나지 않으며 별도의 에러메시지 없이 다시 사용자 입력(>>)을 받을 준비를 한다.

**(4) 멤버 확인: *is_element [집합이름] [원소]***

- 특정 원소가 해당 집합의 원소인지 확인하기 위한 명령어로, 해당 집합의 원소이면 1을 반환하고 그렇지 않으면 0을 반환한다.

```
>> show setA
setA : 1 2 3 4 5
>> is_element setA 2
1
>> is_element setA 8
0
```

**(5) 원소추가: *add [집합이름] [원소]***

- 특정 집합에 원소 하나를 추가한다. 단, 이미 포함되어 있는 원소는 추가 되지 않는다.
- 원소가 추가 될 때, 오름차순으로 정렬된 순서가 되도록 추가해야 한다.

```
>> show setA
```

```
setA : 1 2 3 4 5
>> add setA -4
>> show setA
setA : -4 1 2 3 4 5
>>
```

**(6) 원소 삭제: *pop [집합이름] [원소]***

- 특정 집합에서 해당 원소가 있으면 삭제한다. 해당 원소가 없는 경우는 아무 일도 일어나지 않는다.

```
>> show setA
setA : -4 1 2 3 4 5
>> pop setA 3
>> show setA
setA : -4 1 2 4 5
>> pop setA 10
>> show setA
setA : -4 1 2 4 5
>>
```

- 원소가 하나밖에 없는 집합에서 원소를 삭제 하면 공집합이 된다. 집합 자체가 사리지는 것은 아니다.

**(7) 집합 삭제: *clear [집합이름]***

- 해당 집합을 삭제한다. 해당 집합의 모든 원소와 함께 해당 집합도 삭제되며, 동적할당 받은 메모리들이 할당 해제 되도록 구현한다.

```
>> show setA
setA : -4 1 2 3 4 5
>> clear setA
>> show setA
>>
```

**(8) 프로그램 종료: *quit***

- 프로그램을 종료한다. 종료 시에 동적할당 받은 모든 메모리를 free 시킨 후 종료한다.

```
>> quit
계속 하려면 아무 키나 누르십시오 . . .
```

**(9) 프로그램 실행 예시**

➢ 프로그램 실행은 다음과 같이 실행 한다.

```
C:\>assn5.exe input.txt
```

- 파일명을 입력하지 않은 경우와 입력된 파일이 존재하지 않는 경우는 별도의 에러 메시지 없이 아래와 같이 사용자 입력을 받을 준비를 한다.

```
>>
```

- 파일명이 입력 된 경우, 입력된 파일로부터 명령어를 읽어 실행한다.
  (파일명의 최대 길이는 30자이며, 파일명에는 공백이 없다고 가정한다.)
  파일에는 아래의 예시와 같이 위에서 설명한 명령어가 한 줄에 하나씩 주어진다. 사용자 입력 대신에 순서대로 읽어서 실행하면 된다.

input.txt 예시

```
set sample1 3 5 3 2
add sample1 7
add sample1 8
pop sample1 3
show sample1
set sample2 0
show
add sample2 7
is_element sample2 71
add sample2 7
add sample2 71
is_element sample2 71
show
```

아래는 위 파일을 실행한 예시로 순서대로 명령어 수행 후, 사용자 입력을 위해 기다린다.

```
C:\> assn5.exe input.txt
sample1 : 2 5 7 8
sample1 : 2 5 7 8
sample2 :
0
1
sample1 : 2 5 7 8
sample2 : 7 71
>>
```

아래는 파일명 없이 실행 한 경우의 예시에 해당한다.

```
C:\> assn5.exe
>> set AAA 0
>> add AAA 2
>> add AAA 1
>> add AAA 2
>> add AAA 5
>> add AAA 4
>> add AAA 3
>> show AAA
AAA : 1 2 3 4 5
>> is_element AAA 2
1
>> pop AAA 2
>> is_element AAA 2
0
>> add AAA 7
>> show AAA
```

```
AAA : 1 3 4 5 7
>> pop AAA 3
>> add AAA 2
>> show AAA
AAA : 1 2 4 5 7
>> clear AAA
>> show AAA
>> set AAA 3 13 11 11
>> show AAA
AAA : 11 13
>> quit
계속 하려면 아무 키나 누르십시오 . . .
```

**(힌트)**

- 파일 입출력 & standard I/O: C 언어에서 모니터로 출력하는 standard output과 키보드로 입력 받는 standard input 역시 파일로 처리한다. 따라서 입출력 파일 대신 stdin 혹은 stdout을 사용하면, 입력과 출력을 키보드와 모니터로 할 수 있다.
  아래의 예시를 직접 실행해 보자.

```c
int data;
fprintf(stdout, "정수 입력: ");
fscanf(stdin, "%d", &data);
fprintf(stdout, "%d\n", data);
```

**(헤더 파일 작성)**

- 헤더 파일 작성 예시 (1) (set.h)

```c
#ifndef USER_H
#define USER_H

// 구조체 정의
…
// 함수 선언
…

#endif
```

- 헤더 파일 작성 예시 (2) (set.h)

```c
#pragma once
// 구조체 정의
…
// 함수 선언
…
```

위의 사용자 정의 헤더 파일 set.h를 필요한 곳에서 include 하여 사용한다. 헤더 파일 작성시 위와 같이 작성을 하는 이유에 대해서 간략하게 조사하여 보고서에 서술할 것. (과제 점수에 포함됨)

## ■ Problem 2

**(문제)**

정수를 원소로 가지는 두 집합 A 와 B에 대하여 합집합, 교집합, 차집합 등의 기능들을 수행하도록 작성해 보자.

**(주의사항)**

- *Problem 1* 에서 구현한 프로그램을 기반으로 작성한다. (*Problem 1* 에서 구현한 함수들을 이용할 수 있으며, 이외의 필요한 함수를 추가 구현하여 사용할 수 있다.)

**(설명 및 요구사항)**

- 두 집합은 오름차순으로 정렬된 Linked List를 이용한다. 정렬된 Linked List로 집합이 구현되어 있을 경우, 빠르게 집합에 대한 operation을 수행할 수 있다.

**(1) 초기화면**

- 프로그램을 실행하면 아래와 같이 명령어 입력을 대기하는 ">> "표시가 출력된다. 이 때 사용자는 명령어를 입력 할 수 있다.

```
>>
```

- 구현해야 할 명령어는 6개로 각 명령어의 입력 형식은 아래와 같다.
- **union, intersection, difference**의 입력 형식은
  "명령어 [set A] [set B] [set C]" 이다.
  - set A와 set B는 각각 현재 존재하는 집합의 이름이며, set C는 현재 존재하지 않는, 새로이 만들어질 집합의 이름이다.
  - union, intersection, difference 명령어 입력 시, 새로 생성될 Set C가 이미 있는 집합의 이름이거나 set A 또는 set B가 존재하지 않는 집합의 이름으로 같이 입력된 경우, 아무 일도 일어나지 않으며 별도의 에러메시지 없이 다시 사용자 입력(>>)을 받을 준비를 한다
- **is_disjoint, is_subset**의 입력 형식은
  "명령어 [set A] [set B]" 이다.
  - set A와 set B는 각각 현재 존재하는 집합의 이름이다.
  - is_disjoint와 is_subset 명령어 입력 시, set A 또는 set B가 존재하지 않는 집합의 이름으로 같이 입력된 경우, 아무 일도 일어나지 않으며 별도의 에러메시지 없이 다시 사용자 입력(>>)을 받을 준비를 한다.
- 위 명령어와 함께 입력되는 집합 이름들은 공백 문자로 구분하여 입력한다.

**(2) *union A B C***

- 집합 A와 B의 합집합을 구하여, 그 합집합을 원소로 가지는 새로운 집합 C를 생성한다. 두 개의 집합의 원소들이 정렬되어 있다는 사실을 이용하여 합집합을 구한다. (집합 C의 원소

들도 오름차순 정렬이 되어 있어야 한다.)

```
>> show A
A : 1 2 3 4 5
>> show B
B : 4 5 6 7 8 9
>> union A B C
>> show C
C : 1 2 3 4 5 6 7 8 9
>>
```

**(3) *intersection A B C***

- 집합 A와 B의 교집합을 구하여, 그 교집합을 원소로 가지는 새로운 집합 C를 생성한다. 두 개의 집합의 원소들이 정렬되어 있다는 사실을 이용하여 교집합을 구한다.

```
>> show
A : 1 2 3 4 5
B : 4 5 6 7 8 9
>> intersection A B C
>> show C
C : 4 5
>>
```

- 교집합의 결과가 공집합인 경우, 공집합 C가 생성된다.

**(4) *difference A B C***

- 집합 A에는 속하나 B에는 속하지 않는 차집합을 구하여, 그 차집합을 원소로 가지는 새로운 집합 C를 생성한다. 두 개의 집합의 원소들이 정렬되어 있다는 사실을 이용하여 차집합을 구한다.

```
>> show
A : 1 2 3 4 5
B : 4 5 6 7 8 9
>> difference A B C
>> show C
C : 1 2 3
>>
```

**(5) *is_disjoint A B***

- 집합 A와 B가 서로소(두 집합의 교집합이 공집합)이면 1을 반환하고 그렇지 않으면 0을 반환한다.

```
>> show
A : 1 2 3 4 5
B : 4 5 6 7 8 9
C : 1 2 3
>> is_disjoint B C
1
>> is_disjoint A C
0
>>
```

**(6) *is_subset A B***

- 집합 A의 모든 원소가 집합 B에 속한다면 1을 반환하고 그렇지 않으면 0을 반환한다.

```
>> show
A : 1 2 3
B : 0 1 2 3
C : 1 2 3 4 5 6 7 8 9
>> is_subset A C
1
>> is_subset B C
0
>>
```

**(7) 프로그램 실행 예시**

- 아래는 사용자 입력을 통한 실행 예시이다.

```
C:\> assn5.exe
>> set setA 5 0 3 5 8 90
>> set setB 4 1 2 8 90
>> set setC 10 11 12 13 14 15 16 17 18 19 20
>> show
setA : 0 3 5 8 90
setB : 1 2 8 90
setC : 11 12 13 14 15 16 17 18 19 20
>> difference setA setB setDiff
>> show setDiff
setDiff : 0 3 5
>> intersection setA setC setInter
>> show setInter
setInter :
>> union setB setC setUnion
>> show setUnion
setUunion : 1 2 8 11 12 13 14 15 16 17 18 19 20 90
>> show
setA : 0 3 5 8 90
setB : 1 2 8 90
setC : 11 12 13 14 15 16 17 18 19 20
setDiff : 0 3 5
setInter :
setUnion : 1 2 8 11 12 13 14 15 16 17 18 19 20 90
>> is_disjoint setB setC
1
>> is_subset setA setB
0
>> quit
계속 하려면 아무 키나 누르십시오 . . .
```

- 다음은 파일 읽기를 통해 명령어를 받아 처리하는 예시이다.

input2.txt

```
set sample1 4 10 20 30 40
set sample2 5 10 20 30 40 50
show
union sample2 sample1 sample3
show sample3
is_subset sample2 sample1
is_subset sample1 sample2
difference sample3 sample1 sample4
show sample4
```

아래는 위의 input2.txt 파일에 대한 실행 예시로 다음과 같이 동작 및 출력이 이루어진다.

```
C:\> assn5.exe input2.txt
sample1 : 10 20 30 40
sample2 : 10 20 30 40 50
sample3 : 10 20 30 40 50
0
1
sample4 : 50
>>
```