# CSED101. Programming & Problem solving

# Fall, 2015

## Programming Assignment #3

## (70 points)

**차호준(hersammc@postech.ac.kr)**

- ***Due***: **2015.11.15 23:59**
- ***Development Environment***: **GNU C Compiler (GCC) and Vi Editor (Editor is optional)**

- ***제출물***
  - **C Code files (assn3_1.c, assn3_2.c)**
    - 프로그램의 소스 코드를 이해하기 쉽도록 반드시 **주석**을 붙일 것.
  - **보고서 파일** (.doc(x) or .hwp) 예) assn3.doc(x) 또는 assn3.hwp
    - AssnReadMe.pdf 를 참조하여 작성할 것.
    - **리눅스 서버에 접속하는 것부터 시작해서 프로그램 컴파일 및 실행하는 과정까지를 화면 캡쳐하여 보고서에 포함시키고 간단히 설명 할 것!!**
  - 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.
  - 각 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.)
  - 추가 기능 구현에 대한 점수는 없습니다.
  - **과제 작성시 전역변수는 사용할 수 없으며, 요구되는 기능을 사용자 정의 함수로 적절히 분리하여 코드를 작성하여야 합니다. (main() 함수만 사용하는 경우 감점이 됩니다.)**

## ■ Main Problem: 결! 합! 게임

### 1. 문제

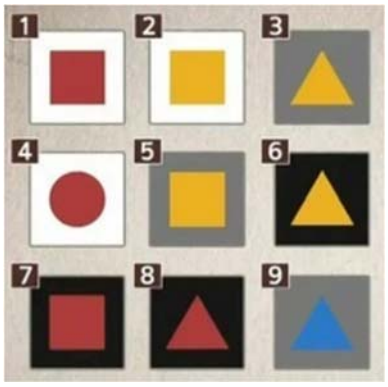
결합 게임은 최근 모 예능 프로그램에 나오면서 유명해진 "Set"이라는 보드게임을 응용한 퍼즐 게임이다. 이 Assignment에서는 2차원 배열을 활용하여 결합 게임판이 주어졌을 때 합이 되는 순서쌍을 모두 찾는 프로그램을 작성해보도록 하자.
(참고: http://picks.be/check/tZUjFKHI9Y 온라인 결합게임)

### 2. 목적

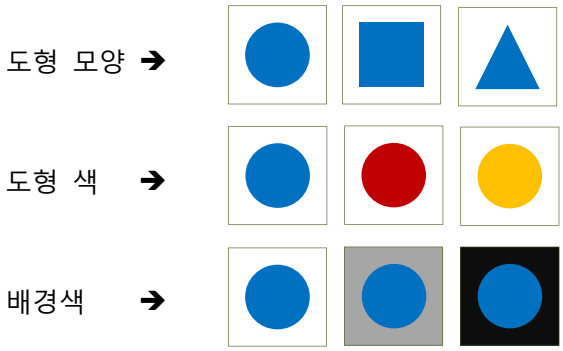
- 2차원 배열의 선언과 사용을 익힌다. (2차원 배열 이상 사용 가능)
- 함수에서 2차원 배열을 매개변수로 사용하는 방법을 익힌다.
- 텍스트 파일 입출력을 익힌다.

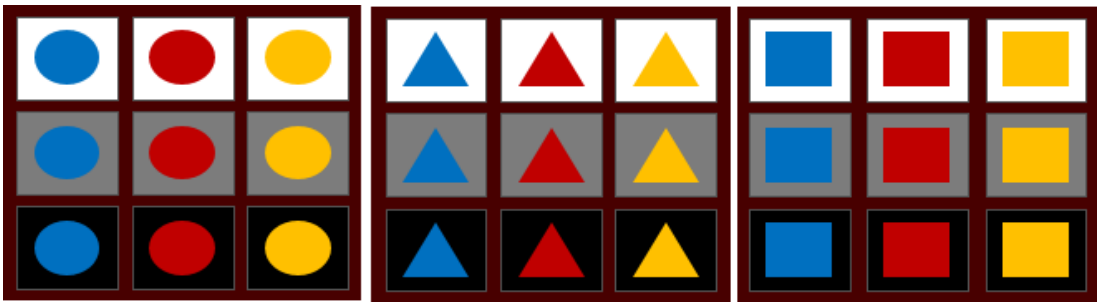
### 3. Rule

결합게임은 9장의 그림 중 그림의 속성이 모두 같거나 모두 다른 조건 3가지를 가진 그림 3장을 찾아내는 게임이다. 우선 결합게임은 다음과 같은 3 X 3으로 되어있는 게임판을 이용한다.
결합판에서는 가장 왼쪽 위 그림에서부터 가로 방향으로 3개씩, 아래 방향으로 3줄에 숫자를 부여하여 각각 1 ~ 9번이라고 부른다.



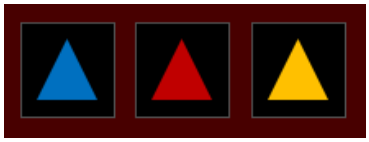

게임에 사용되는 그림은 도형 모양, 도형 색, 배경색의 3가지 속성을 갖고 있으며, 각 속성은 3가지의 형태로 등장한다. 이 3 가지 속성이 각각 다르게 조합된 27장의 그림으로 게임이 진행된다.





**<세 가지 속성이 조합된 27장의 그림>**

게임판에는 27장의 그림 중 9장의 그림이 공개되며, 이 중 합이 되는 3장의 그림을 모두 찾는 것이 이 게임의 목표이다. '합'이란 3장의 그림의 속성 각각이 모두 같거나 모두 다른 그림들로 이루어진 3장의 그림을 말한다.

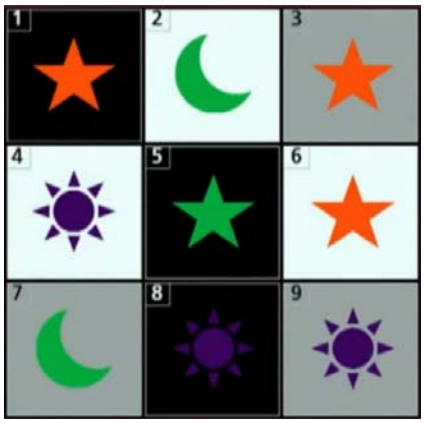


예를 들어, 위 3장의 그림은 도형 모양이 모두 같고, 도형 색이 모두 다르고, 배경색이 모두 같아 '합'이 된다. 아래 3장의 그림은 도형 색이 모두 다르고, 배경색이 모두 같지만 도형의 모양이 동그라미 2개, 세모 1개로 이루어져 있으므로 '합'이 아니다.

공개된 9장의 그림에서 '합'을 모두 찾아 더 이상 '합'이 없다고 판단되면 '결'을 외친다. '결'을 외쳤을 때, '합'이 되는 그림이 더 이상 없을 경우, 해당 게임판으로 진행하는 게임은 종료된다. (더 자세한 룰은 다음 링크의 영상을 참조 할 것: https://youtu.be/0gCfLVFs79w )

**4. Algorithm**

위의 규칙만 알고 응용하면 결합 게임에서 '합'이 되는 조합을 모두 찾을 수 있다. 모 예능 프로그램에서 나온 '합'이 되는 조합 찾는 법을 소개하면 다음과 같다.




**2-1. 바탕색이 같은 그림들을 비교한다.**

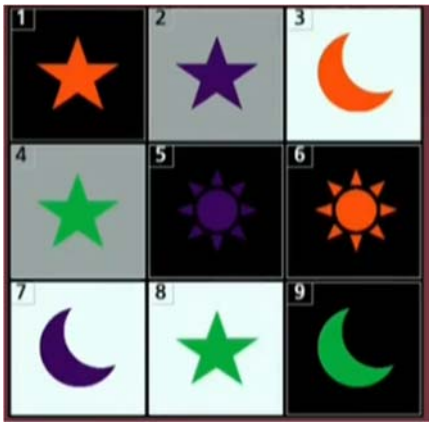
위의 결합판을 예로 들어보자. 배경색이 검은색으로 같은 1, 5, 8을 비교한다. 그러나 별이 2개,

해가 1개 있으므로 '합'이 되지 않는 것을 알 수 있다. 하지만 배경색이 회색으로 같은 3, 7, 9와 배경색이 흰색으로 같은 2, 4, 6은 '합'이 되는 것을 알 수 있다.

**2-2. 도형 모양이 같은 그림들을 비교한다.**

이 때 이미 2-1에서 '합'을 확인한 바탕색이 같은 그림들은 제외하고 생각한다.

도형 모양이 별로 같은 1, 3, 5, 6을 비교한다. 여기서 1, 3, 6이 '합'이 되는 것을 알 수 있다. 또한, 해로 모양이 같은 4, 8, 9가 '합'이 되는 것을 알 수 있다.




2-3. **도형 색이 같은 그림들을 비교한다.**

이 때 이미 위에서 확인한 바탕색과 모양이 같은 그림들은 제외하고 생각한다.

바로 위의 결합판 그림에서 도형 색이 보라색으로 같은 2, 5, 7을 비교하면 배경색도 다 다르고, 모양도 다 달라 '합'이 되는 것을 알 수 있다. 그러나 도형 색이 주황색으로 같은 1, 3, 6은 배경색이 같은 그림이 있기 때문에 '합'이 아닌 것을 알 수 있다.

**2-4. 모든 속성이 다른 그림들을 찾는다.**

먼저 숫자가 가장 적은 배경색을 찾는다. 위의 결합판에서는 회색 배경색이 된다. 그리고 그 다음으로 배경색 숫자가 적은 것을 찾는다. 위의 예에서는 흰색 배경색이 된다. 회색 배경색과 흰색 배경색에서 모양과 색이 다른 그림을 선택한 후에, 남은 검은색 배경에서 모든 속성(도형 모양과 도형색)이 다른 것을 찾는 방식이다.

예를 들어 회색 배경색에서 4번(녹색 별)을 선택한 경우, 모양과 색이 다 다른 흰색 배경색은 3번(주황 달)과 7번(보라 달)이다.

(1) 회색 4번(녹색 별) 선택 후, 흰색 3번(주황 달) 선택한 경우, 선택된 2개의 그림과 합이 되는 그림(검은색 배경에 보라 해)을 찾는다. 해당하는 그림이 5번에 있으며 3, 4, 5는 '합'임을 알 수 있다.

(2) 회색 4번(녹색 별) 선택 후, 흰색 7번(보라 달) 선택한 경우, 선택된 2개의 그림과 합이 되는 그림(검은색 배경에 주황 해)를 찾는다. 해당하는 그림이 6번에 있으며 4, 6, 7은 '합'임을 알 수 있다.

글로 이해하기 어려우신 분들은 영상으로 된 설명 http://tvcast.naver.com/v/491278 를 참조해주시기 바랍니다.

혹은 더 쉬운 알고리즘이 있다면 직접 생각해서 구현하셔도 됩니다.

## ■ Problem 1: 결합 판 생성 (20 점)

**(문제)**

결합 게임에 사용할 결합 판을 직접 생성하는 프로그램을 작성해 보자. 랜덤으로 결합 판을 생성한 후, 그 정보를 텍스트 파일로 저장한다.

**(설명 및 요구사항)**

1. 프로그램을 실행하면, 아래와 같이 생성할 결합 판의 개수를 입력하라는 메시지가 출력된다.

```
[hersammc@programming Assn3]$ ./assn3_1
생성할 결합 판의 개수:
```

사용자는 생성하고자 하는 결합 판의 개수를 입력한다. 단, 잘못된 입력은 없다고 가정한다.
(입력 가능한 범위: 양의 integer 값 안에서 모두 가능 함.)

2. 사용자가 입력한 숫자만큼 결합판이 생성되어 파일에 저장된다. 저장할 파일의 이름은 "**problem.txt**"로 한다. (NOTE: 이름이 다를 시 감점)
   아래는 생성할 결합 판의 개수를 2개로 입력한 경우에 생성된 예시 파일의 내용이다.

**첫 번째 결합판**

**두 번째 결합판**

```
2
#[B .{A #[A
%<A %{C %<C
%[B .[B #<A

#{B %{C .[B
#<B .{C .{B
%{A .[C #{A
```

(1) 파일의 맨 윗줄에는 생성할 결합 판의 개수를 저장한다.
(2) 2번째 줄부터는 결합 판의 정보를 생성하여 저장한다.
(위의 파일 예시로 설명을 하면 2번째 줄부터 5번째 줄까지가 하나의 결합판을 나타낸다.)
(3) 각 줄은 3개의 그림 정보를 가지도록 하며, 하나의 그림 정보는 총 3개의 연속된 문자로 구성한다. 각 그림 정보는 1개의 공백으로 구분을 해서 저장하도록 하자.
(4) 하나의 그림 정보는 순서대로 배경 색 1글자, 도형 색 1글자, 도형 모양 1글자로 총 3글자(3개의 char)로 이루어져 있으며, 각 속성은 아래와 같다.
- **배경 색**: '.', '#', '%'
- **도형 색**: '<', '{', '['
- **도형 모양**: 'A', 'B', 'C'

(5) 하나의 결합 판은 9개의 그림을 가지게 되는데, 이 때 9개의 그림은 서로 다른 그림이어야 한다. 즉, 하나의 결합 판에 배경 색, 도형 색, 도형 모양이 모두 같은 그림은 존재할 수 없다.
(6) 각각의 결합 판은 한 줄을 띄워서 구분하여 저장한다.
(7) 결합 판은 매 실행 시마다 다르게 생성되어 저장되도록 구현한다.

**(주의사항)**

1. 확장자를 포함한 소스 파일 이름은 "assn3_1.c"로 저장할 것.
2. 보고서는 assn3.doc(x) 또는 "assn3.hwp"로 저장할 것. (보고서는 통합으로 작성)
3. 출력은 출력 예시와 동일하도록 작성할 것. (모든 spacing은 space로 작성)
4. 아직 수업에서 다루지 않은, Array를 제외한 Data Structure 등은 사용하지 말 것.

## ■ Problem 2: 결! 합! 게임 (50 점)

**(문제)**

실제로 결합 게임을 하는 프로그램을 작성하자.

게임을 위해 각 결합 판의 정보를 파일에서 읽는다. 이 때, 입력예시로 제공한 "problem.txt"와 Problem 1 에서 생성한 파일 모두에서 프로그램이 실행되어야 한다. (마지막 페이지에, 어싸인 문서와 함께 제공한 "problem.txt" 파일에 대한 합의 목록이 있으니 참고할 것.)

**(설명 및 요구사항)**

1. 프로그램을 실행하면, 간단히 게임에 대한 설명을 출력 한 후 파일 "problem.txt"에서 정보를 읽어 아래의 실행예시와 같이 출력한다.

<"problem.txt" 예시>

```
2
#[B .{A #[A
%<A %{C %<C
%[B .[B #<A

#{B %{C .[B
#<B .{C .{B
%{A .[C #{A
```

<실행예시>

```
[hersammc@programming Assn3]$ ./assn3_2
합일 경우 1 2 3 과 같이 입력해주시고, 결일 경우 -1을 입력해주세요.
단, 합은 오름차순으로 써야합니다. 1 2 3은 답으로 처리하지만 3 1 2는 오답으로 처리합니다.
합이 맞을 경우 +1점, 결이 맞을 경우 +3점, 틀릴 경우 -1점입니다.

====================== 1 Round / 2 Round ======================

현재 점수 : 0

+---+---+---+
|###|...|###|
|[B]|{A}|[A]|
|###|...|###|
+---+---+---+
|%%%|%%%|%%%|
|<A>|{C}|<C>|
|%%%|%%%|%%%|
+---+---+---+
|%%%|...|###|
|[B]|[B]|<A>|
|%%%|...|###|
+---+---+---+

합 :
```

(1) 파일의 첫 번째 줄에는 게임의 총 Round 수가 주어진다.

(2) 파일의 두 번째 줄부터는 각 Round의 결합 판 정보가 주어진다.

(3) 각 줄은 3개의 그림 정보를 가지며, 하나의 그림 정보는 순서대로 배경 색 1글자, 도형 색 1글자, 도형 모양 1글자로 총 3글자(3개의 char)로 이루어져 있으며, 각 속성은 아래와 같다.

- **배경 색**: '.', '#', '%'
- **도형 색**: '<', '{', '['
- **도형 모양**: 'A', 'B', 'C'

2. 각 라운드의 시작 시, "problem.txt"에서 결합 판의 정보를 읽어와 현재 Round 정보, 현재 점수를 출력한 후, 결합 판을 출력한다.

(1) 각 그림은 3 X 3 사이즈로 출력한다.

(2) 각 그림의 첫 줄과 마지막 줄은 배경색을 출력한다.

(3) 각 그림의 중간 줄은 (도형 색)(도형 모양)(도형 색 반대 괄호) 순으로 출력한다. 예를 들어 도형 모양이 'B', 도형 색이 '[', 배경색이 '#'인 경우 오른쪽 표와 같이 출력한다.
(힌트) ASCII 코드 상으로 '<', '>', '[', ']', '{', '}' 의 값의 확인

| | | |
|---|---|---|
| # | # | # |
| [ | B | ] |
| # | # | # |

(4) 각 그림은 –와 | (or 연산자 2개짜리 중 1개)로 구분하며, 모서리는 +로 구분한다.

(5) 라운드 시작의 출력문은 다음을 사용한다. (*변수1, 2, 3*은 각자 코드에 맞출 것.)

```
printf("======================== %d Round / %d Round
========================₩n₩n", 변수1, 변수2);
printf("현재 점수 : %d₩n₩n", 변수3);
```

3. 현재 점수와 결합 판을 출력 한 후, 사용자로부터 합이 되는 그림 조합을 입력 받는다. 단, 중복 번호(1 1 2), 음수 입력 등 잘못된 입력은 없다고 가정한다.

(1) 각 그림의 번호는 오른쪽 표와 같다.

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |

(2) 사용자는 합이라고 생각하는 그림의 번호를 3개 입력한다. 3개의 번호는 1 2 6과 같이 한 칸씩 공백으로 구분하여 입력하며 오름차순으로 입력한다. 즉, 1 2 6이 합인 경우, 6 1 2를 입력하면 오답으로 처리한다.

- 입력 받은 조합이 합인 경우, 현재 점수에서 1점을 얻는다.
- 입력 받은 조합이 합이 아닌 경우, 현재 점수에서 1점을 잃는다.
- 입력 받은 조합이 이미 합을 한 조합일 경우, 현재 점수에서 1점을 잃는다.
- 위의 첫 번째 결합판에 대한 사용자의 입력 예시는 아래와 같다.

```
합 : 1 2 6
합이 맞습니다. +1점 (현재 : 1점)
합 : 1 7 8
합이 맞습니다. +1점 (현재 : 2점)
합 : 4 5 7
합이 맞습니다. +1점 (현재 : 3점)
합 : 2 4 9
합이 아닙니다. -1점 (현재 : 2점)
합 : 2 3 4
합이 맞습니다. +1점 (현재 : 3점)
합 : 1 7 8
이미 나온 합입니다. -1점 (현재 : 2점)
```

(3) 사용자는 현재 결합판에서 더 이상 합이 되는 그림 조합이 없다고 판단되면 즉, 결이라고 판단되면 아래의 예시처럼 0을 입력한다.

- 합이 되는 그림 조합이 더 이상 없어 '결'일 경우 현재 점수에 3점을 받는다.
- 합이 되는 그림 조합이 아직 남아 '결'이 아닌 경우 현재 점수에서 1점을 잃는다.
- 위의 첫 번째 결합판에 대한 사용자의 입력 예시는 다음과 같다.

```
합 : 0
결에 실패했습니다. -1점 (현재 : 1점)
합 : 5 8 9
합이 맞습니다. +1점 (현재 : 2점)
합 : 0
결에 성공했습니다. +3점 (현재 : 5점)
현재 점수 : 5
```

**(힌트: 결의 성공여부를 알기 위해서는 합이 될 수 있는 것이 무엇인지 모두 알아야 한다. )**

4. 결이 성공한 경우, 다음 라운드로 넘어간다.

```
현재 점수 : 5

====================== 2 Round / 2 Round ======================

현재 점수 : 5

+---+---+---+
¦###¦%%%¦...¦
¦{B}¦{C}¦[B]¦
¦###¦%%%¦...¦
+---+---+---+
¦###¦...¦...¦
¦<B>¦{C}¦{B}¦
¦###¦...¦...¦
+---+---+---+
¦%%%¦...¦###¦
¦{A}¦[C]¦{A}¦
¦%%%¦...¦###¦
+---+---+---+

합 :
```

5. 모든 라운드가 종료된 경우, 아래와 같이 게임 종료 메시지와 함께 최종 점수를 출력한다.

```
합 : 0
결에 성공했습니다. +3점 (현재 : 11점)
현재 점수 : 11

게임을 종료합니다.
최종 점수 : 11

[hersammc@programming Assn3]$
```

**(주의사항)**

1. 확장자를 포함한 소스 파일 이름은 "assn3_2.c"로 저장할 것.
2. 보고서는 assn3.doc(x) 또는 "assn3.hwp"로 저장할 것. (보고서는 통합으로 작성)
3. 출력은 출력 예시와 동일하도록 작성할 것. (모든 공백은 space로 작성)
4. 아직 수업에서 다루지 않은, Array 를 제외한 Data Structure 등은 사용하지 말 것.

■ 결합판 예시와 합

(1)
(2)
(3)
(4)
(5)

```
5
#[B .{A #[A
%<A %{C %<C
%[B .[B #<A

#{B %{C .[B
#<B .{C .{B
%{A .[C #{A

%{A .[C % [B
#{C #[C %{C
#[B %<B #{B

%{A %{B %<B
#[A %[A #{C
.{B #<C %[B

#[C % [A #<A
%<B %{C .[B
.<C #{B .{A
```

| (1) 결합판 | (2) 결합판 | (3) 결합판 | (4) 결합판 | (5) 결합판 |
|---|---|---|---|---|
| 1 2 6 | 1 5 7 | 합이 없음 | 1 6 7 | 1 2 6 |
| 1 7 8 | 2 6 9 | | 2 3 9 | 1 3 8 |
| 2 3 4 | 4 7 8 | | 5 7 8 | 1 4 9 |
| 4 5 7 | | | | 1 5 7 |
| 5 8 9 | | | | 2 3 9 |
| | | | | 2 4 5 |
| | | | | 2 7 8 |
| | | | | 3 4 7 |
| | | | | 3 5 6 |
| | | | | 4 6 8 |
| | | | | 5 8 9 |
| | | | | 6 7 9 |