CSED101. Programming & Problem solving
Fall, 2019

Programming Assignment #2
(60 points)

이지완 (jiwan95@postech.ac.kr)

- ***Due***: **2019.10.18 23:59**
- ***Development Environment***: **Windows Visual Studio 2019**

- ***제출물***
  - **C Code files (*.c)**
    - 확장자 포함 소스 파일명은 해당 문제 번호를 따라 assn2_1.c, assn2_2.c로 할 것
      - 파일명 양식을 미준수하거나 프로젝트 폴더를 통째로 제출할 시 감점
    - 프로그램의 소스 코드를 이해하기 쉽도록 반드시 주석을 붙일 것
    - 함수 또는 변수 이름 선택 시, 어떠한 의도로 사용되는지 명료하게 나타낼 수 있도록 할 것
  - **보고서 파일 (assn2.docx or assn2.hwp)**
    - AssnReadMe.pdf 를 참조하여 작성할 것
    - 프로그램 실행 화면을 캡처하여 보고서에 포함시키고, 간단히 설명할 것
    - **명예 서약(Honor code)**: 표지에 다음의 내용을 포함한다. "나는 이 프로그래밍 과제를 다른 사람의 부적절한 도움 없이 완수하였습니다." 보고서 표지에 명예 서약이 없는 경우는 과제를 제출하지 않은 것으로 처리
    - 소스코드와 보고서 파일을 LMS으로 제출

- ***주의사항***
  - 각 문제에 해당하는 요구사항을 반드시 지킬 것
  - **컴파일 또는 실행이 안되면 무조건 0점 처리**
  - **하루 late시 20%가 감점되며, 3일 이상 지나면 받지 않음 (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.)
  - 추가적인 기능 구현에 대해서는 추가 점수 없음

■ Problem 1: 흑백 공 게임 (40점)

(목적)
이번 과제는 랜덤 함수 사용과 조건문, 반복문 사용을 익히고자 한다. 또한 사용자 정의 함수들을 작성하여 코드의 재사용을 높이는 연습을 한다.

(주의사항)

- 함수를 정의하고 사용하는 방법을 익히는 문제이므로 **main 함수에 모든 기능을 구현한 경우 0점 처리함**.
- 과제에서 요구하는 사용자 정의 함수들은 파라미터 개수 또는 파라미터 자료형, 함수 이름, 리턴 자료형 등 자유롭게 변경 가능함. 그러나 **동일한 기능을 하는 함수는 반드시 있어야 하며, 변경 시 무엇을 어떻게 변경해서 구현했는지 보고서에 포함**하도록 한다. 이 이외에 필요하다면 다른 함수들로 더 나누거나 추가 가능함.
- **전역 변수, 배열 및 goto 문은 사용할 수 없으며, 사용 시 0점 처리함.**
- 문제의 출력 형식은 실행 예시와 동일하게 작성 바람.

0. 게임 설명

이 게임은 두 명의 플레이어가 각자에게 주어진 15개의 흰색 공을 검은색 공으로 바꾸어 가는 게임이다. 종료 시점 시 검은 공의 수가 더 많은 플레이어가 승리하며, 게임이 진행되는 동안 플레이어들은 번갈아가며 자신의 턴을 진행한다. 한 플레이어가 종료를 선언하면 다른 플레이어는 자동으로 자신의 턴 한 번을 진행하게 되며, 최대 세 번까지 추가로 진행할 수 있다.
매 턴마다 플레이어는 1부터 5까지의 랜덤한 값으로 흰색 공을 검은색 공으로 바꿀 수 있다. 만약 검은색 공이 15개를 초과할 경우, 초과한 수만큼의 검은색 공만 갖는다. 예를 들어, 한 플레이어가 13개의 검은색 공을 갖고 있으며, 이 상태에서 1과 2의 수를 기대하며 턴을 진행한다. 이때 2 초과의 수인 4가 나온다면 해당 플레이어는 2개의 검은색 공과 13개의 흰색 공을 갖게 된다.

1. 프로그램 초기 화면

프로그램을 실행하면, 아래 실행 예시와 같이 각 플레이어에게 주어진 15개의 흰색 공이 출력되며 플레이어의 입력을 기다린다.

```
P1 ○○○○○○○○○○○○○○○
P2 ○○○○○○○○○○○○○○○

PROCEED OR NOT? ENTER ANSWER WITH [GO 3, END –PLAYER NUM(-1 OR -2)]
```

2. 플레이어 입력

플레이어는 매 턴마다 입력 값을 제공하며, 유효한 입력 값은 3, -1, -2 가 있다. 유효한 입력 값이

들어올 경우 다음과 같은 시나리오로 게임이 진행된다.

A. 게임 계속 진행 (입력 값 3)

플레이어 1과 플레이어 2가 둘다 게임 진행을 원하면 입력 값으로 3을 입력한다. 이 경우 플레이어 1과 2가 순서대로 검은색 공을 받게 된다. 검은색 공의 수는 컴퓨터가 1개부터 5개까지를 랜덤하게 선택하여 제공한다.

다음은 입력 값 3의 예시이다 (사용자 입력 값 빨간색 밑줄 표시). 플레이어 1이 먼저 4개의 흰색 공을 검은색 공으로 바꾸었으며, 이후 플레이어 2가 흰색 공 2개를 검은색으로 바꾸었다. 이때 2와 4는 컴퓨터로부터 랜덤하게 받은 값이다.

```
P1 ○○○○○○○○○○○○○○○○
P2 ○○○○○○○○○○○○○○○○

PROCEED OR NOT? ENTER ANSWER WITH [GO 3, END –PLAYER NUM(-1 OR -2)] 3

AFTER P1
P1 ●●●●○○○○○○○○○○○○
P2 ○○○○○○○○○○○○○○○○

AFTER P2
P1 ●●●●○○○○○○○○○○○○
P2 ●●○○○○○○○○○○○○○○

PROCEED OR NOT? ENTER ANSWER WITH [GO 3, END –PLAYER NUM(-1 OR -2)]
```

플레이어 1의 게임 진행

플레이어 2의 게임 진행

입력 값 3 예시: 게임 진행

게임 진행 중 검은색 공이 15개를 초과할 경우, 초과한 수만큼의 검은색 공만 갖는다. 다음 예시에서 두 플레이어 모두 검은색 공이 12개인 상태에서 3을 입력하여 게임을 진행한다. 플레이어 1의 경우 검은색 공 2개를 더 받아 14개가 되었으나, 플레이어 2의 경우 5개를 더 받게 되어 검은색 공 수가 15개를 초과하게 되었다. 이 경우, 게임 규칙에 따라 플레이어 2는 초과한 수만큼의 검은색 공 2개만 갖게 됨을 볼 수 있다.

```
AFTER P2
P1 ●●●●●●●●●●●○○○
P2 ●●●●●●●●●●●○○○

PROCEED OR NOT? ENTER ANSWER WITH [GO 3, END –PLAYER NUM(-1 OR -2)] 3

AFTER P1
P1 ●●●●●●●●●●●●●○
P2 ●●●●●●●●●●●○○○
```

AFTER P2
P1 ●●●●●●●●●●●●●○
P2 ●●○○○○○○○○○○○○○

PROCEED OR NOT? ENTER ANSWER WITH [GO 3, END –PLAYER NUM(-1 OR -2)]

15개 초과 예시

다음은 과제에서 요구하는 함수들이다. 동일한 기능을 하는 함수는 반드시 있어야 하며, 변경 시 보고서에 설명을 추가하도록 한다 (주의사항 참고).

- 플레이어로부터 입력 값을 받아 그 값을 반환하는 함수로, 입력을 요구하는 출력 문구를 포함하여 구현한다.
  int getResponse()

- 플레이어가 랜덤한 값의 검은색 공을 받는데 사용하는 함수이다. 랜덤한 값의 범위를 지정할 수 있도록 시작과 끝을 매개변수로 전달받는다. 이 값은 매 실행마다 다른 값을 가져 예측 불가능해야 한다.
  int getRandomBalls(int start, int end)

- 플레이어의 검은색 공 보유 수를 계산하여 반환하는 함수이다. 현재 갖고 있는 검은색 공 수와 추가되는 검은색 공 수를 매개변수로 전달받아, 15개가 넘지 않도록 검은색 공 수를 반환한다.
  int changeBallStatus(int current, int plus)

- 플레이어의 흑백 공 보유 상태 출력을 위한 함수로, 검은색 공의 수를 매개변수로 전달받아 화면에 출력한다. 흰색 공은 ○, 검은색 공은 ●을 이용하여 출력한다 (복사 붙여넣기).
  void printCurrentBallStatus(int num)

- 다음 두 함수들은 각 플레이어의 게임을 진행한다. 플레이어 1과 2의 검은색 공 개수를 전달받아, 각 함수마다 해당하는 플레이어의 변경된 검은색 공의 수를 반환한다. (입력 값 3 예시 참고)
  int movePlayer1(int player1, int player2)
  int movePlayer2(int player1, int player2)

B. 게임 종료 선언 (입력 값 -1, -2)

입력 값 -1과 -2는 해당 플레이어 번호의 게임 종료 선언을 의미한다. 둘 중의 한 플레이어가 게임 종료 선언을 하면, 상대 플레이어는 게임을 지속할지 말지를 결정하며 최대 세 번까지 본인의 턴을 추가로 진행할 수 있다.
다음은 입력 값 -1의 예시이다 (사용자 입력 값 빨간색 밑줄 표시). 입력 값 -1은 플레이어 1의 종료 선언을 의미하며, 플레이어 2가 자동으로 본인의 턴을 한 번 진행한다. 이후 플레이어 2도

-2를 입력하여 게임 종료를 선언하였다. 두 플레이어 모두가 종료를 선언하였으므로 승패 출력문과 함께 게임은 종료된다.

```
AFTER P2
P1 ●●●●●●●●●●●●●○
P2 ●●○○○○○○○○○○○○○

PROCEED OR NOT? ENTER ANSWER WITH [GO 3, END –PLAYER NUM(-1 OR -2)] -1

AFTER P2
P1 ●●●●●●●●●●●●●○
P2 ●●●●○○○○○○○○○○

PROCEED OR NOT? ENTER ANSWER WITH [GO 3, END –PLAYER NUM(-1 OR -2)] -2

WIN P1, P1 14, P2 4
```

입력 값 -1 예시

다음은 입력 값 -2의 예시이다 (사용자 입력 값 빨간색 밑줄 표시). 입력 값 -2는 플레이어 2의 종료 선언을 의미하며, 플레이어 1이 자동으로 본인의 턴 한 번을 진행한다. 이 예시는 최대 세 번까지인 본인의 추가 턴들을 모두 진행한 경우이며, 승패 출력문과 같이 프로그램은 종료된다.

```
AFTER P2
P1 ●●○○○○○○○○○○○○○
P2 ●●●●●●●●●●●●●○○

PROCEED OR NOT? ENTER ANSWER WITH [GO 3, END –PLAYER NUM(-1 OR -2)] -2

AFTER P1
P1 ●●●●●●●○○○○○○○○
P2 ●●●●●●●●●●●●●○○

PROCEED OR NOT? ENTER ANSWER WITH [GO 3, END –PLAYER NUM(-1 OR -2)] 3

AFTER P1
P1 ●●●●●●●●●○○○○○
P2 ●●●●●●●●●●●●●○○

PROCEED OR NOT? ENTER ANSWER WITH [GO 3, END –PLAYER NUM(-1 OR -2)] 3

AFTER P1
P1 ●●●●●●●●●●●●●○○○
P2 ●●●●●●●●●●●●●○○
```

```
WIN P2, P1 12, P2 13
```

입력 값 -2 예시

C. 유효하지 않은 입력

3, -1, -2 이외의 입력 값들은 유효하지 않다. 유효하지 않은 입력을 받은 경우, 입력 받은 값이 유효하지 않다는 문구가 출력되며 프로그램은 종료된다. 이외에 플레이어 종료 선언이 중첩될 경우 입력 값이 유효하지 않은 경우로 처리한다. 이 문제에서는 플레이어 입력으로 정수 외 입력을 고려하지 않으므로 예외 처리를 구현할 필요는 없다.
다음 예시들은 각각 유효하지 않은 입력이 들어온 경우와 중복된 종료 선언의 경우를 보여준다 (사용자 입력 값 빨간색 밑줄 표시).

```
AFTER P2
P1 ●●○○○○○○○○○○○○○
P2 ●●●●●○○○○○○○○○○

PROCEED OR NOT? ENTER ANSWER WITH [GO 3, END –PLAYER NUM(-1 OR -2)] 0
0 IS NOT VALID RESPONSE

WIN P2, P1 2, P2 5
```

3, -1, -2 이외의 입력 예시

```
AFTER P2
P1 ●●○○○○○○○○○○○○○
P2 ●●●●●●●●●○○○○○

PROCEED OR NOT? ENTER ANSWER WITH [GO 3, END –PLAYER NUM(-1 OR -2)] -1

AFTER P2
P1 ●●○○○○○○○○○○○○○
P2 ●●●●●●●●●●●●●○○

PROCEED OR NOT? ENTER ANSWER WITH [GO 3, END –PLAYER NUM(-1 OR -2)] -1
-1 IS NOT VALID RESPONSE

WIN P2, P1 2, P2 13
```

중복된 종료 선언 예시

다음은 과제에서 요구하는 함수이다. 동일한 기능을 하는 함수는 반드시 있어야 하며, 변경 시 보고서에 설명을 추가하도록 한다 (주의사항 참고).

- 현재 입력 값이 유효한지 확인하는 함수로, 현재 입력 값과 이전 입력 값을 매개변수로 받는다. 이전 입력 값은 종료 선언이 중복되었는지 확인하기 위해 사용되며, 이 변수는 직전 입력 값을 의미하지 않는다. 현재 입력 값이 유효하다면 참을 나타내는 1을, 유효하지 않다면 거짓을 나타내는 0을 반환한다.
  int isResponseValid(int response, int prevResponse)

3. 승패 결정

게임은 승패 출력문과 함께 종료한다. 종료 시점 때 두 플레이어들 중 검은색 공 수가 많은 플레이어가 승리하며, 같을 경우 무승부가 된다. 다음은 세 가지 경우의 승패 출력문을 보여주는 예시들이다.

```
AFTER P2
P1 ●●●●●●●●●●●●●○
P2 ●●●●●●●●●●●○○○○

PROCEED OR NOT? ENTER ANSWER WITH [GO 3, END –PLAYER NUM(-1 OR -2)] -1

AFTER P2
P1 ●●●●●●●●●●●●●○
P2 ●●●●●●●●●●●●●○○

PROCEED OR NOT? ENTER ANSWER WITH [GO 3, END –PLAYER NUM(-1 OR -2)] -2

WIN P1, P1 14, P2 13
```

게임 종료 예시: 플레이어 1 우승

```
AFTER P2
P1 ●●●●●●●●●●●●●○○
P2 ●●●●●●●●●●●●●●○

PROCEED OR NOT? ENTER ANSWER WITH [GO 3, END –PLAYER NUM(-1 OR -2)] -2

WIN P2, P1 13, P2 14
```

게임 종료 예시: 플레이어 2 우승

```
AFTER P2
P1 ●●●●●●●●●●●●●○
P2 ●●●●●●●●●●●●●●○

PROCEED OR NOT? ENTER ANSWER WITH [GO 3, END –PLAYER NUM(-1 OR -2)] 0
0 IS NOT VALID RESPONSE
```

```
DRAW, P1 14, P2 14
```

게임 종료 예시: 무승부

다음은 과제에서 요구하는 함수들이다. 동일한 기능을 하는 함수는 반드시 있어야 하며, 변경 시 보고서에 설명을 추가하도록 한다 (주의사항 참고).

- 두 플레이어들의 게임 승패를 판단하여 그 승패를 판단하는 함수이다. 두 플레이어의 검은색 공 수를 매개변수로 전달받아 승리한 플레이어의 번호 또는 무승부를 의미하는 0을 반환한다.
  int calcWinner(int player1, int player2)

- 게임의 승패 결과에 따라 출력 문구를 달리하여 출력하는 함수이다. 승패 결과는 함수 내에서 calcWinner() 함수를 호출하여 얻는다. 출력문은 승패 결과, 플레이어 1의 검은색 공의 수, 플레이어 2의 검은색 공의 수 순으로 출력하며, 위 예시와 동일하게 작성한다.
  void printWinner(int player1, int player2)

■ Problem 2: 볼링 점수 계산 (20점)

(목적)
이번 과제는 문제 이해를 바탕으로 조건문, 반복문, 사용자 정의 함수 사용을 목적으로 한다.

(주의사항)

- 함수를 정의하고 사용하는 방법을 익히는 문제이므로 **main 함수에 모든 기능을 구현한 경우 0점 처리함**.
- 과제에서 요구하는 사용자 정의 함수는 파라미터 개수 또는 파라미터 자료형, 함수 이름, 리턴 자료형 등 자유롭게 변경 가능함. 그러나 **동일한 기능을 하는 함수는 반드시 있어야 하며, 변경 시 무엇을 어떻게 변경해서 구현했는지 보고서에 포함**하도록 한다.
- 이번 문제에서는 요구하는 사용자 정의 함수가 하나이므로, **사용자 정의 함수 두 개를 더 추가**하도록 한다. 위 설명과 동일하게 추가하고자 하는 함수들의 이름, 기능 등은 자유롭게 구현한다.
- **전역 변수, 배열 및 goto 문은 사용할 수 없으며, 사용 시 0점 처리함.**
- 입력 값은 파일로부터 읽어와야 하며, 사용자로부터 입력 받을 경우 감점 처리함.
- 문제의 출력 형식은 실행 예시와 동일하게 작성 바람.

0. 규칙 설명

볼링 게임 점수 계산하는 프로그램을 작성한다. 볼링 게임의 결과는 사전에 작성된 파일로부터 읽어와야 하며, 기호와 숫자를 해석하여 점수 계산을 한다. 점수 계산 규칙은 다음과 같다.

1. 한 게임은 총 10개의 프레임으로 구성되어 있다.
2. 각 프레임마다 볼링 핀 10개를 쓰러트리는 게임이며 볼링 핀 1개당 1점이다.
3. 각 프레임은 두 번의 시도를 할 수 있다.
   A. 볼링 핀 10개를 첫 번째 시도로 모두 쓰러트릴 경우 스트라이크(S)라 한다. 이 경우 두 번째 시도는 사라진다. 단, 마지막 프레임에서는 스트라이크를 치더라도 사라지지 않으며, 결과적으로 해당 프레임에서는 세 번의 기회가 주어진다.
   B. 볼링 핀 10개를 두 번의 시도로 모두 쓰러트릴 경우 스페어(P)라 한다. 마지막 프레임에서 스페어를 칠 경우 한 번의 시도가 추가적으로 주어지며, 결과적으로 해당 프레임에서는 세 번의 기회가 주어진다.
4. 다음의 경우에는 추가 점수를 받을 수 있다.
   A. 스트라이크를 치면 그 다음 두 번의 시도 동안 쓰러트린 볼링 핀들의 개수만큼 추가 점수를 받는다.
   B. 스페어를 치면 그 다음 시도 때 쓰러트린 볼링 핀의 개수만큼 추가 점수를 받는다.
   C. 마지막 프레임에서는 스트라이크, 스페어 상관없이 추가 점수를 받을 수 없다.

다음은 점수 계산 예시이다. 점수판에 스트라이크는 S, 스페어는 P, 핀을 하나도 못 쓰러트린

경우는 –으로 표시된다. 마지막 프레임을 제외한 모든 프레임은 두 시도의 결과 값으로 나타낼 수 있으며, 마지막 프레임은 스페어나 스트라이크가 있을 경우 세 번의 시도 값으로 기입된다. 표내 사선 표시는 첫 시도로 스트라이크를 칠 경우 다음 시도는 사라짐을 의미한다. 점수 계산 예시의 프레임별 점수는 게임의 결과로써 주어지는 값이며, 이번 과제는 게임 결과의 점수 계산을 목표로 한다. (파란색 표시)

프레임		1	2	3	4	5	6	7	8	9	10
점수		8 1	7 P	9 -	S	6 3	S	8 P	S	S	6 P S
점수계산	기본	9	10	9	10	9	10	10	10	10	20
	추가	0	9	0	9	0	10	10	16	10	0
	누적 합계	9	28	37	56	65	85	105	131	151	**171**

점수 계산 예시

다음은 점수 계산 예시의 프레임별 계산 과정을 나타낸다. 기본 점수는 추가 점수를 고려하지 않은 점수로, 쓰러트린 볼링 핀의 개수이다. 프레임 내 각 시도별 점수를 정수 값으로 해석한 값들로 계산 가능하다. 각 프레임마다 스페어 또는 스트라이크를 칠 경우, 기본 점수는 최대 값인 10을 갖는다 (마지막 프레임 제외). 추가 점수는 규칙에 따라 프레임 내 결과 값들에 스페어나 스트라이크가 있을 경우 계산한다.

프레임 1. 기본 점수는 8 + 1 = **9**이다.

프레임 2. 기본 점수는 7 + 3 = 10이다. 스페어를 친 경우, 다음 프레임의 첫 번째 시도 값인 9만큼 추가 점수를 받는다. 따라서 프레임 2의 합계는 **19**가 된다.

프레임 3. 기본 점수는 9 + 0 = **9**이다.

프레임 4. 기본 점수는 10이다. 스트라이크를 친 경우 다음 프레임의 두 시도들의 값을 추가 점수를 받으며, 이 경우 6 + 3 = 9이다. 따라서 프레임 4의 합계는 **19**가 된다.

프레임 5. 기본 점수는 6 + 3 = **9**이다.

프레임 6. 기본 점수는 10이다. 스트라이크를 친 경우 다음 프레임의 두 시도들의 값을 추가 점수를 받으며, 이 경우 8 + 2 = 10이다. 따라서 프레임 6의 합계는 **20**이 된다.

프레임 7. 기본 점수는 8 + 2 = 10이다. 스페어를 친 경우, 다음 프레임의 첫 번째 시도 값인 10만큼 추가 점수를 받는다. 따라서 프레임 7의 합계는 **20**이 된다.

프레임 8. 기본 점수는 10이다. 스트라이크를 친 경우 다음 프레임의 두 시도들의 값을 추가 점수를 받으며, 이 경우 10 + 6 = 16이다. 따라서 프레임 8의 합계는 **26**이 된다.

프레임 9. 기본 점수는 10이다. 스트라이크를 친 경우 다음 프레임의 두 시도들의 값을 추가 점수를 받으며, 이 경우 6 + 4 = 10이다. 따라서 프레임 9의 합계는 **20**이 된다.

프레임 10. 기본 점수는 6 + 4 + 10 = 20이다. 마지막 프레임은 추가 점수를 계산하지 않으므로 프레임 10의 합계는 **20**이 된다.

1. 볼링 결과 입력

볼링 게임 결과가 기록되어 있는 텍스트 파일에서 문자들을 읽어 볼링 점수를 계산한다. 다음은

볼링 결과 입력 예제들이며, 과제2 공지 란에서 텍스트 파일들을 받을 수 있다 (Example1-4). 각 시도마다 쓰러트린 볼링 핀의 개수가 공백없이 주어지며, 위 점수 계산 예시와 동일하게 스트라이크는 S, 스페어는 P, 핀을 하나도 못 쓰러트린 경우는 –으로 표시된다.

Example1.txt

```
9-8P72S9P-9S-P9-SS8
```

Example2.txt

```
SSSSSSSSSSS
```

Example3.txt

```
9P8-SS8-S819P81S9P
```

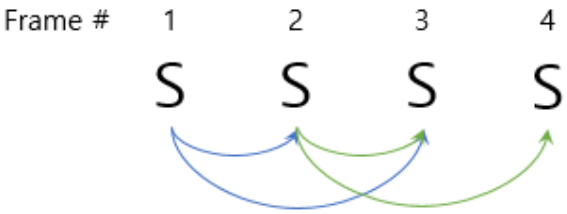
Example4.txt

```
6P71SS9PS8PS9P9-
```

각 예제들의 결과 값은 순서대로 150, 300, 155, 182이다. 두 번째 예제는 모든 시도들에서 스트라이크를 친 경우이며, 결과 값인 300은 받을 수 있는 점수의 최대 값이다. 이는 스트라이크 한 번이 30점으로 (현재 기본 점수 10점에 다음 프레임 추가 점수 10점, 그 다음 프레임 추가 점수 10점) 계산될 수 있기에 가능한 점수이다.
입력 파일의 볼링 결과 값은 사용자로부터 받아오는 값이 아니므로 추가적인 예외 처리 구현은 필요하지 않다. 즉, 비정상적인 결과 값에 대해서는 고려할 필요가 없다. 예시 파일로 4개의 파일이 주어지나, 이 과제에서는 점수 계산을 위하여 읽는 텍스트 파일 이름을 Example.txt로 고정해서 구현하도록 한다.

2. 점수 계산

기본 점수는 추가 점수를 고려하지 않은 점수로, 쓰러트린 볼링 핀 개수마다 1점씩 계산된다. 볼링 결과가 입력된 파일은 위 결과 입력 설명과 같이 기호와 숫자로 이루어져 있다.
추가 점수는 스트라이크나 스페어를 칠 경우 추가로 얻을 수 있는 점수이다. 스페어는 다음 한 번 기회 동안의 기본 점수를 추가로 얻을 수 있으며, 스트라이크는 두 번 동안의 기본 점수를 추가로 얻을 수 있다. 이는 해당 문자 이후 각각 한 번(P), 두 번(S) 동안 기본 점수를 추가 점수로 더해줘야 한다는 의미이다.
다음은 스트라이크 네 번이 연속된 경우이며, 프레임 1과 프레임 2의 스트라이크로부터 필요한 추가 점수를 나타내고 있다. 프레임 2의 스트라이크는 1번, 프레임 3의 스트라이크는 2번 (프레임 1 스트라이크로부터 한 번, 프레임 2 스트라이크로부터 한 번)으로 기본 점수가 추가 점수로 참조된 것을 알 수 있다.



스트라이크 네 번 연속된 경우

다음은 과제에서 요구하는 함수이다. 동일한 기능을 하는 함수는 반드시 있어야 하며, 변경 시 보고서에 설명을 추가하도록 한다. 이외에 사용자 정의 함수 두 개를 추가적으로 구현하도록 한다 (주의사항 참고).

- 기호와 숫자 문자를 해석하는 함수이다. 해석이 필요한 문자와 이미 해석된 이전 문자의 정수 값을 매개변수로 전달받으며, 해석된 정수 값을 반환하는 함수이다. 이전 문자의 정수 값은 P(스페어)를 해석하는데 이용될 수 있다.
  int encode(char c, int prevPoint)

3. 결과 출력

결과 출력은 다음과 같이 볼링 결과 입력 문자들, 최종 계산 점수 순으로 출력한다.

```
C:₩WINDOWS₩system32₩cmd.exe
9-8P72S9P-9S-P9-SS8
TOTAL SCORE: 150
```

Example1 출력 결과