CSED101. Programming & Problem solving

Spring, 2014

Programming Assignment #3

(70 points)

김우중 (woojoong@postech.ac.kr)

- ***Due*: 2014. 05. 15 23:59**
- ***Development Environment:* GNU C Compiler (GCC) and Vi Editor (Editor is optional)**

- ***제출물***
  - **C code files (*.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.
  - 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 제출할 것. 그 외의 다른 이름으로 제출하면 감점 또는 0점 처리된다.
  - 과제를 작성하는 데 있어서 전역변수를 선언하여 사용할 수 없다.
  - 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.)
  - 이번 과제에서는 추가 기능 구현에 대한 추가 점수는 없습니다.
  - 이번 과제의 주제는 배열입니다. 다음 과제의 주제가 포인터이므로 이번 과제에서는 포인터를 사용할 수 없으며, 사용할 시에는 감점이 됩니다.

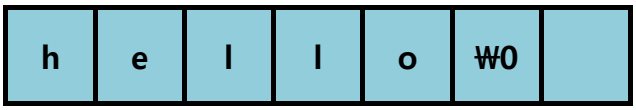
Problem 1: 행맨 게임 (Hangman Game) (30점)

[들어가기 전]

1. 문자열(string)
   - 연속된 문자들로 C 언어에서 문자열 앞 뒤에 " "를 이용한다.
   - char 형의 1차원 배열을 이용하여 문자열을 저장한다.
   - 배열에 문자열을 저장할 때는 끝을 NULL 문자 ('₩0')를 넣어서 표시한다.

2. 선언
   - char str[] = "hello";
     위와 같이 선언과 동시에 초기화를 하게 되면, 자동으로 문자열의 끝에 널문자가 추가된다.

     str

   - str[0] == 'h', str[1] == 'e', str[2] == 'l', str[3] == 'l', str[4] == 'o', str[5] == '₩0'와 같이 배열처럼 사용할 수 있다.

3. 입력과 출력

```
char str[100];
printf("Inut your povis id: ");
scanf("%s", str);
printf("%s", str);
```

<결과화면> (아래의 파란 글자는 사용자 입력입니다.)

```
Input your povis id: woojoong
woojoong
```

[문제]

행맨 (Hangman) 게임은 상대방이 생각하는 단어를 맞춰가는 게임이다. 컴퓨터가 선택한 단어를 맞춰가는 행맨 게임을 작성해 보자.

[가정 및 요구사항]

① 단어의 최대길이는 20 이라고 가정한다.
#define 지시자를 이용하여 단어의 최대길이를 정의하여 사용한다.

② 단어는 모두 영문 소문자로만 구성되어 있다.

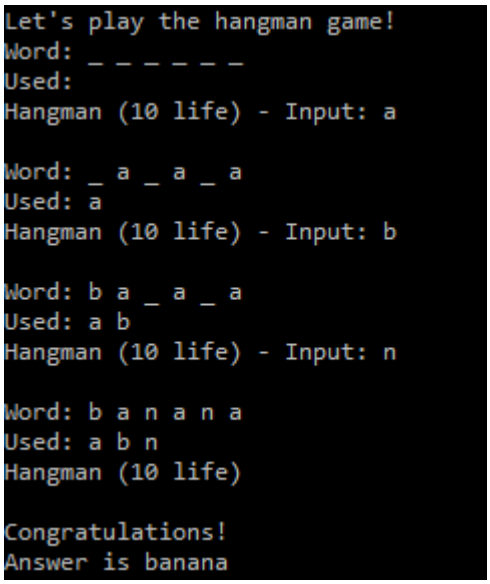
③ 모든 입력 값은 영문 소문자만 입력 받는다고 가정한다.

④ 전역변수를 선언하여 사용할 수 없다.

[설명]

컴퓨터가 dic.txt 파일에 있는 단어 중 하나를 랜덤하게 선택한다. 그리고 아래의 예시와 같이 선택된 단어의 길이만큼 '_'가 출력된다. 행맨의 생명(life)은 10 부터 시작한다. 단어를 맞추기 위해 사용자가 영문자를 하나 입력하는데, 선택된 단어에 입력한 영문자가 없는 경우 life 의 수가 1 감소한다. 10번의 없는 문자를 입력하여 life 가 0이 되기 전에 단어를 맞춰야 하며 life 가 0 이 되면, 선택된 단어를 공개(출력)하고 게임은 종료된다.
선택된 단어에 입력 받은 문자가 있는 경우, 아래의 예시와 같이 해당 부분의 문자만 출력한다.
사용자가 입력한 모든 문자는 Used란에 입력한 순서대로 출력한다. 단, 입력한 문자가 Used란에 있는 경우, life를 감소하지 않고 이미 입력된 값이라고 출력한다.

아래의 예제는 게임이 실행되어, banana 라는 단어가 랜덤하게 선택된 경우이다.



■ **단어 선택**

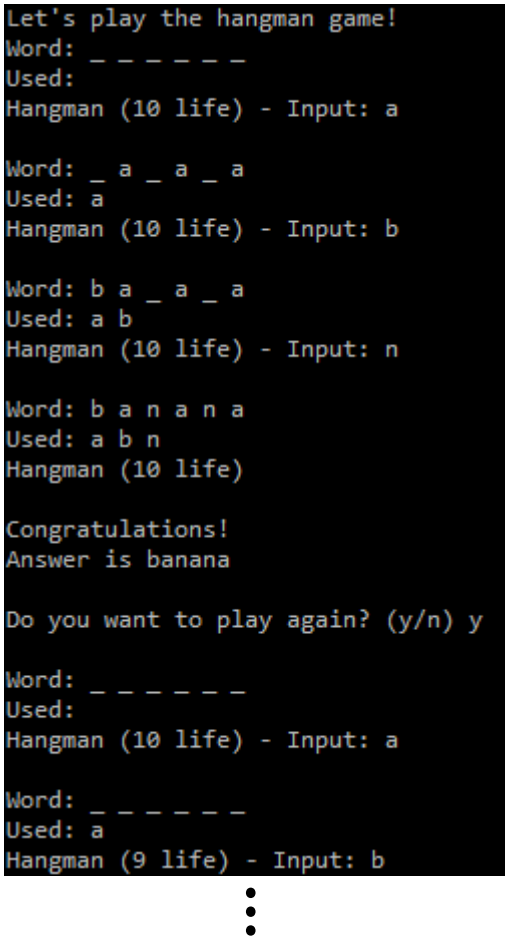
① 파일 dic.txt 의 구성은 아래와 같다.

```
5
apple
banana
cat
dictionary
flower
```

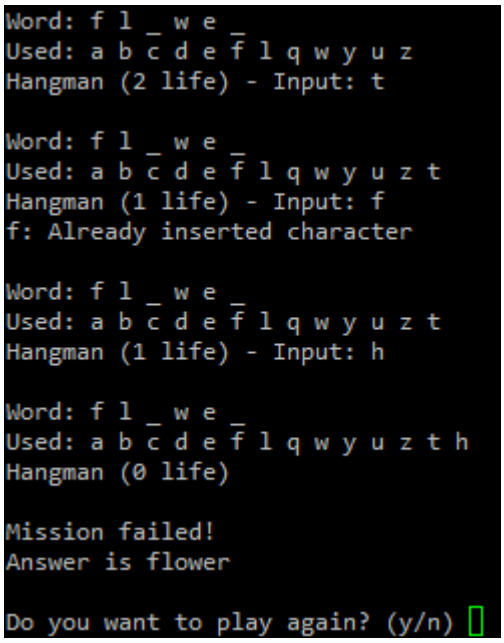
- 첫 번째 줄의 숫자는 전체 단어의 수를 나타낸다.
- 나머지 데이터들은 단어이며 줄(line)단위로 구분된다.
- 위의 파일 내용은 하나의 예제이며, 위와 같이 구성된 어떤 파일에 대해서도 처리할 수 있도록 구현해야 한다.

② 프로그램을 실행하면, 위의 파일에서 단어 1개를 랜덤하게 선택한다.

■ **<u>아래 예시를 참고하여 프로그램을 작성하라.</u>**



⋮



■ **게임 종료**

사용자가 선택된 단어를 life 가 0 이 되기 이전에 맞추거나, life 가 0 이 되면 게임이 종료된다. 게임이 끝나면, 게임을 다시 하겠느냐는 메시지가 출력되며 사용자가 n 을 선택하면 프로그램이 종료된다. 사용자가 y 를 선택하면 파일에서 랜덤하게 단어를 선택한 후 게임을 다시 시작한다.

[주의사항]

- 파일 이름은 **"assn3_1.c"**로 저장 한다.
- 보고서는 **"assn3.doc"** or **"assn3.hwp"**로 저장 한다. (보고서는 통합하여 작성)
- 출력은 실행예시와 동일하게 작성해야 한다.
- 아직 수업시간에 다루지 않은 문법은 사용하지 않는다 (e.g., 포인터).
- string.h에 정의된 함수를 사용하지 않는다.
  - 해당 기능이 필요한 경우 char 배열을 이용, 직접 구현하여 사용한다.
- 기타 문의사항은 LMS 시스템 질의응답 혹은 담당 TA에게 메일을 통해 문의한다.

Problem 2: 슬라이딩 퍼즐 (Sliding puzzle) (40점)

[문제]

슬라이딩 퍼즐을 맞추는 프로그램을 작성해 보자. 슬라이딩 퍼즐은 그림 2-1과 같이 24개의 퍼즐 블록과 하나의 빈 칸으로 구성되어있다. 퍼즐을 맞추기 전 상태가 그림 2-1(a)라고 했을 때, 빈 칸인 *를 이동시켜가면서 그림 2-1(b)의 상태로 만들면 성공한다.

A	B	Q	R	S
U	C	T	P	O
V	D	K	*	L
E	W	J	I	M
F	X	G	H	N

(a) Initial state of sliding puzzle

A	B	C	D	E
F	G	H	I	J
K	L	M	N	O
P	Q	R	S	T
U	V	W	X	*

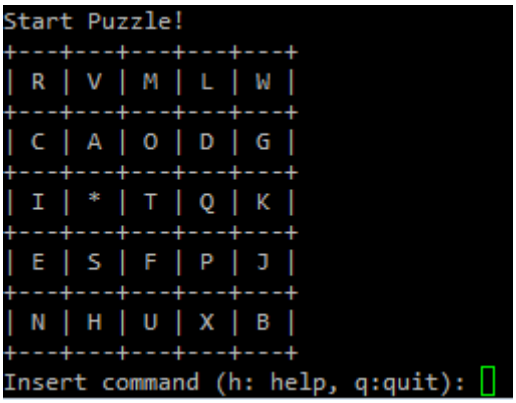
(b) Goal of sliding puzzle

그림 2-1. Examples of sliding puzzle ('*' means empty slot)

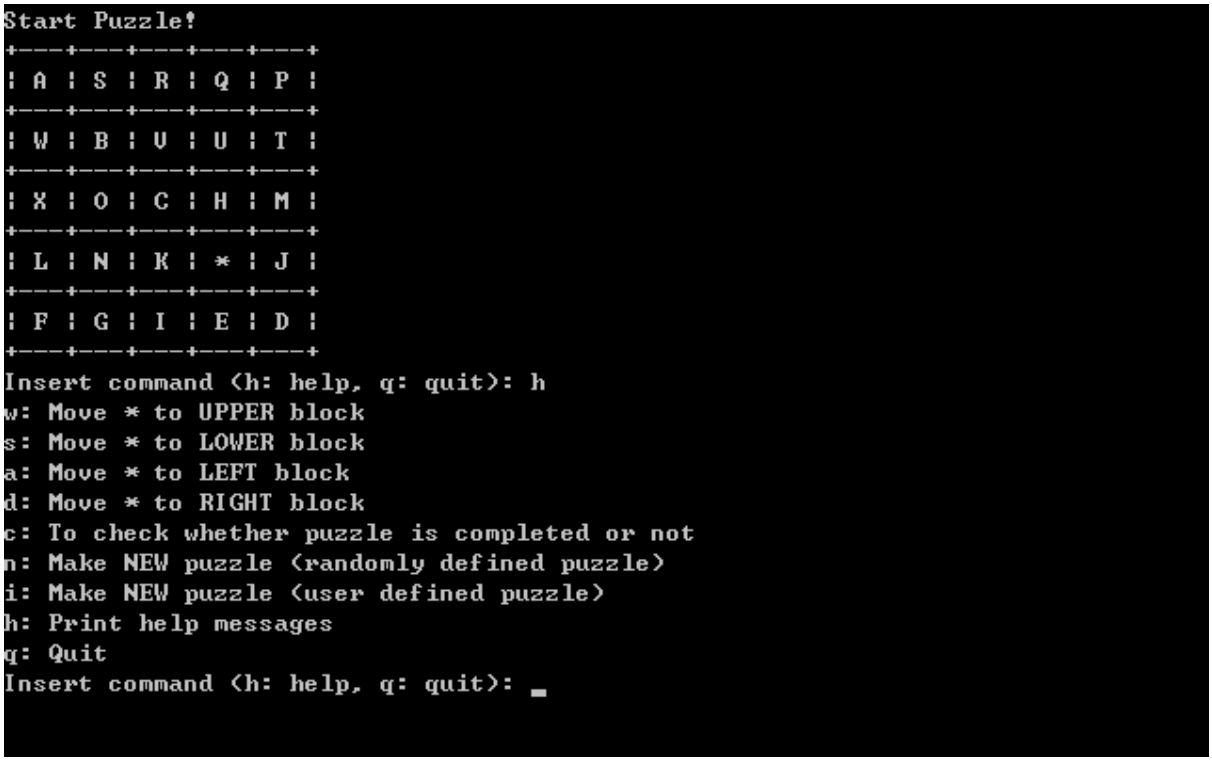
[요구사항]

0. 전역변수를 선언하여 사용할 수 없다.
1. 프로그램을 실행시키면, 배열을 랜덤한 값으로 초기화한다. 이 때, **randomGenerate** 라는 이름의 함수를 정의하여 사용한다.
    ① 5×5 배열을 선언해서 사용한다.
    ② 각 배열의 들어갈 수 있는 값은 A~X 까지(24 개의 알파벳)의 값과, 빈 칸을 의미하는 *이다.
        - 빈 칸은 반드시 한 개만 존재한다.
        - A 부터 X 까지의 값을 채울 때, 서로 중복되는 값이 없어야 하며 모두 대문자여야 한다.

2. 초기 출력 화면은 다음과 같다.
    ① Start Puzzle! 이라는 메시지를 출력한다.
    ② 생성된 Puzzle 을 출력한다. 이 때, **puzzlePrint** 함수를 정의해서 사용한다.
    ③ Command 라인을 출력한다.



3. **h 를 입력**하면, 어떤 명령어를 사용할 수 있는지 아래와 같이 help 메시지를 출력한다.

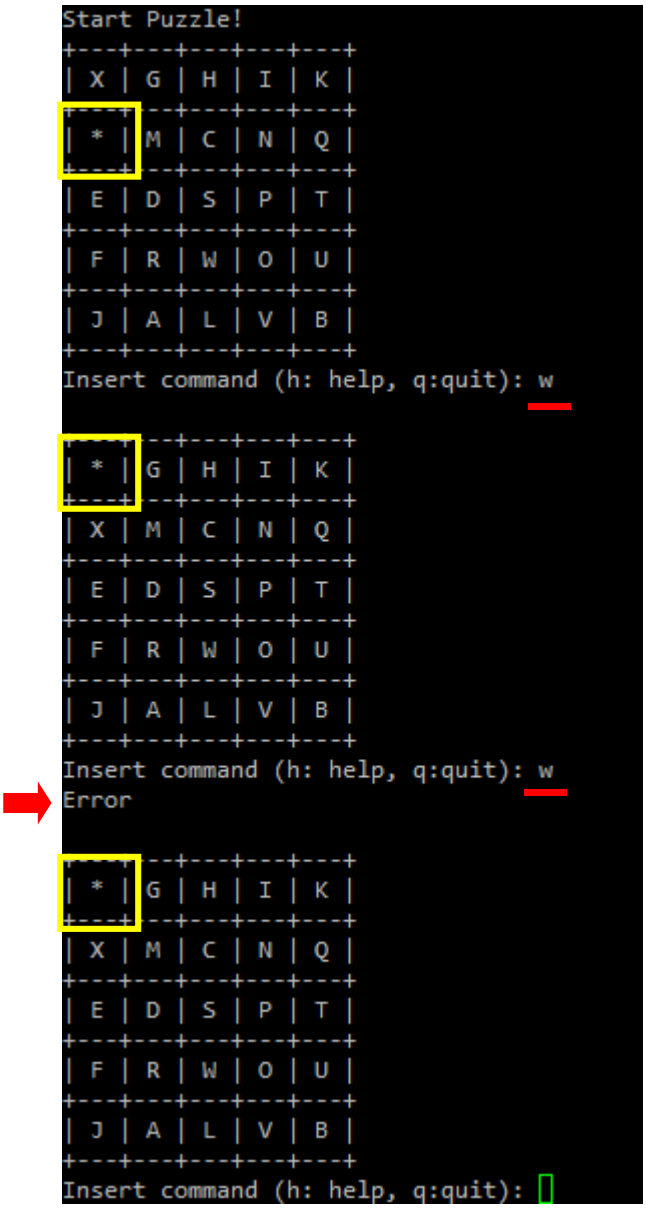


위의 help 메시지에서 볼 수 있듯이 명령어 w, s, a, d, c, n, i, h, q 외의 입력에 대해서는 고려하지 않는다.

4. **q 를 입력**하면 프로그램을 종료한다.

5. **<u>w, s, a, d 키를 입력하면</u>** 빈칸인 *를 위, 아래, 좌, 우로 각각 움직인다. 아래의 사용자 정의 함수를 정의해서 사용한다.

① **moveToUpperBlock:** w 키를 입력하면, 빈칸 *를 한 칸 위로 올린다. 만일 올릴 칸이 없으면 아래와 같이 Error 라는 메시지를 출력한다.



② **moveToLowerBlock:** s 키를 입력하면, 빈칸 *를 한 칸 아래로 내린다. 만일 내릴 칸이 없으면 Error 라는 메시지를 출력한다.

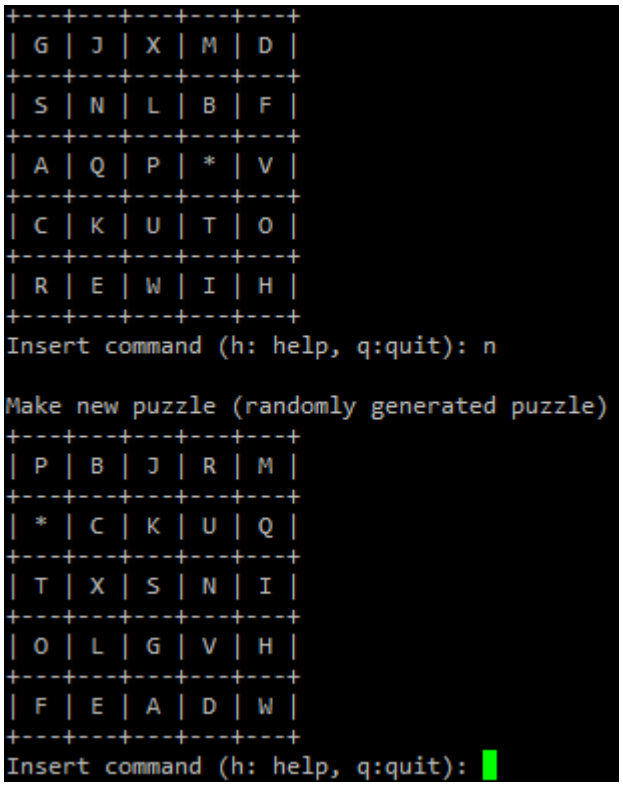
③ **moveToLeftBlock:** a 키를 입력하면, 빈칸 *를 한 칸 왼쪽으로 움직인다. 만일 움직일 칸이 없으면 Error 라는 메시지를 출력한다.

④ **moveToRightBlock:** d 키를 입력하면, 빈칸 *를 한 칸 오른쪽으로 움직인다. 만일 움직일 칸이 없으면 Error 라는 메시지를 출력한다.

6. **c 를 입력하면** 퍼즐이 완성되었는지 확인한다. 이 때, **isCompleted** 함수를 정의해서 사용한다. 그림 2-1(b)이 완성된 상태이다.
   ① 퍼즐이 완성된 상태가 아니면, 계속 진행하라는 메시지 출력과 함께 현재 퍼즐의 상태를 출력한다.
   ② 퍼즐이 완성된 상태이면, 완료라는 메시지와 함께 현재 퍼즐의 상태를 출력한다.

<퍼즐이 완성되지 않은 상태> **<퍼즐이 완성된 상태>**

7. **n 을 입력하면** 새로운 랜덤 퍼즐을 생성한다. 이 때 randomGenerate 라는 함수를 호출한다.



8. **i를 입력하면** 사용자가 퍼즐을 직접 생성한다. 이 때, **inputPuzzle** 함수를 정의해서 사용한다.

```
Insert command (h: help, q:quit): i

Input puzzle element
Puzzle[0][0]: A
Puzzle[0][1]: A
Puzzle[0][2]: A
Puzzle[0][3]: A
Puzzle[0][4]: A
Puzzle[1][0]: A
Puzzle[1][1]: A
Puzzle[1][2]: A
Puzzle[1][3]: A
Puzzle[1][4]: A
Puzzle[2][0]: A
Puzzle[2][1]: A
Puzzle[2][2]: A
Puzzle[2][3]: A
Puzzle[2][4]: A
Puzzle[3][0]: A
Puzzle[3][1]: A
Puzzle[3][2]: A
Puzzle[3][3]: A
Puzzle[3][4]: A
Puzzle[4][0]: A
Puzzle[4][1]: A
Puzzle[4][2]: A
Puzzle[4][3]: A
Puzzle[4][4]: A
Invalid character or duplcated character
Input puzzle element
Puzzle[0][0]: A
Puzzle[0][1]: W
Puzzle[0][2]: X
Puzzle[0][3]: V
Puzzle[0][4]: F
Puzzle[1][0]: B
Puzzle[1][1]: S
Puzzle[1][2]: T
Puzzle[1][3]: U
Puzzle[1][4]: G
Puzzle[2][0]: C
Puzzle[2][1]: R
Puzzle[2][2]: Q
Puzzle[2][3]: H
Puzzle[2][4]: D
Puzzle[3][0]: N
Puzzle[3][1]: O
Puzzle[3][2]: *
Puzzle[3][3]: I
Puzzle[3][4]: E
Puzzle[4][0]: M
Puzzle[4][1]: L
Puzzle[4][2]: K
Puzzle[4][3]: J
Puzzle[4][4]: P
Generated puzzle is
+---+---+---+---+---+
| A | W | X | V | F |
+---+---+---+---+---+
| B | S | T | U | G |
+---+---+---+---+---+
| C | R | Q | H | D |
+---+---+---+---+---+
| N | O | * | I | E |
+---+---+---+---+---+
| M | L | K | J | P |
+---+---+---+---+---+
Insert command (h: help, q:quit):
```

① 사용자 입력이 끝나면, 게임에 유효한 퍼즐인지 체크한다. 이 때, **checkPuzzle** 함수를 정의해서 사용한다.

② 아래와 같이 유효하지 않은 입력 값에 대해 체크 후, 오류 메시지를 출력하고 처음부터 다시 입력 받는다.

- A~X 까지의 문자 혹은 *가 아닌 경우
- 각 문자들이 중복된 경우

③ 오류 메시지

Invalid character or duplicated character

④ 유효한 경우 왼쪽과 같은 메시지 출력 후, 퍼즐이 출력된다.

[주의사항]

- 파일 이름은 **"assn3_2.c"**로 저장 한다.
- 보고서는 **"assn3.doc"** or **"assn3.hwp"**로 저장 한다. (보고서는 통합하여 작성)
- 출력은 실행예시와 동일하게 작성해야 한다.
- 아직 수업시간에 다루지 않은 문법은 사용하지 않는다 (e.g., 포인터).
- 각 명령어 별로 함수를 제작하여 프로그래밍 한다.
  - 제시된 함수의 이름과 달라도 무방하며, 필요하다면 추가적으로 다른 함수를 구현해도 좋다.
- 기타 문의사항은 LMS 시스템 질의응답 혹은 담당 TA에게 메일을 통해 문의한다.