COMP319 Algorithms 1, Spring 2021
Homework Programming Assignment 4 (HW4)
알고리즘1, 2020년 가을학기, 프로그래밍 숙제 4

Instructor: Gil-Jin Jang Email: gjang@knu.ac.kr
School of Electronics Engineering, Kyungpook National University
장길진, 경북대학교 전자공학부

프로그래밍 숙제 4번의 목적은 다음과 같다.

1. 문자열 hashing 과 linked list 를 이용한 collision 해결 (chaining)
2. Hash map 을 alphabet 순서로 유지
3. Hashing 을 이용한 단어의 효율적 검색과 빈도수 세기
4. 단어의 빈도수를 이용한 hash map 정렬 및 encoding

공통 요구사항:

1. `int main(int argc, char *argv[]) { ... }`" 로 main 함수가 작성되어야 한다(주어진 template code 활용).
2. 처음의 주석문에는 본인의 정보가 기재되어야 함.
3. 다른 학생의 코드와 거의 동일하면 COPY 판정을 받을 수 있기 때문에, 본인이 공개된 코드를 참조했으면 웹주소, 책이름, 강의자료 차수와 쪽수 등을 주석문으로 넣는다. 만약 다른 학생의 코드와 유사한데 참조정보가 없으면 COPY 판정을 받게 된다.
4. 한글문자는 코드에 넣지 않는다. 한글 사용 자체에 대한 감점은 없지만 컴파일 오류를 발생시킬 수 있으며, 실행점수를 전혀 못 받을 수도 있음. 이름도 영어로 적는 것을 권장한다.
5. 한글 문자를 주석 등에 사용하고자 하는 경우 linux machine 에서 컴파일해 보고 확인하거나, UTF8 로 인코딩을 바꾸어 제출한다(안전하게는 안 쓰는 편이 좋다).
6. 참고: 두 파일의 차이를 비교할 때에는 windows 에서는 winmerge (`http://winmerge.org`), linux 에서는 vimdiff, meld 사용 가능.

1 Homework 4-1

입력파일예: 다음과 같이 단어들이 나열되어 있으며, 특수문자는 없고 알파벳 대/소문자, 숫자로만 이루어져 있으며 space/linebreak/tab 문자로 단어들은 구분된다(즉, `scanf` 함수로 읽어들이면 된다.)

```
1  FILE: i31.txt (number of words not known)
2  sit lorem dolor diam diam lorem sed dolor diam sit amet
3  diam diam lorem ipsum lorem ipsum amet sit lorem sed
4  ipsum sit ipsum amet dolor dolor sed sit ipsum diam
5
```

```
6  FILE: imagine.txt (number of words not known)
7  imagine there s no heaven it s easy if you try no hell below us above us only sky imagine
       all the people livin for today ah imagine there s no countries it isn t hard to do
       nothing to kill or die for and no religion too imagine all the people livin life in
       peace you you may say i m a dreamer but i m not the only one i hope someday you ll
       join us and the world will be as one imagine no possessions i wonder if you can no
       need for greed or hunger a brotherhood of man imagine all the people sharing all the
       world you you may say i m a dreamer but i m not the only one i hope someday you ll
       join us and the world will live as one
```

할것들: 주어진 텍스트 파일에서 단어를 하나씩 읽어서 chaining 을 이용한 hash table 에 저장한다. 중복되는 단어는 몇 개가 입력되었는지 그 수를 세어 저장하고, 각 bucket의 list는 단어의 오름차순으로 유지한다. 모든 단어를 읽고 난 후에 생성된 hash table 을 텍스트 파일에 출력하고, 전체 단어를 오름차순으로 출력하는 프로그램을 작성한다.

실행예 및 자세한 설명: 채점에 사용되는 command line argument 는 hash map size/입력파일/출력파일이다.

(a) line 1, hash map size (`argv[1]`): positive integer. 본 예제에서는 3

(b) line 1, 입력파일(`argv[2]`): 위에 예제로 보임

(c) line 1, 출력파일(`argv[3]`): 아래에 '`cat`' 명령 이후

(d) line 2, 화면출력줄1: **0.00026 seconds** 측정된 소요시간

(e) line 2, 화면출력줄2: **324 bytes (46.286 x 7)** 측정된 사용 메모리 동적 메모리로 사용된 메모리의 byte 수와 입력크기(7단어)의 배수로 표현됨

(f) line 5: '`cat`' 명령은 텍스트 파일을 화면에 출력함. 즉 '`output/i31_4-1_h3.txt`' 파일의 내용을 보임

(g) line 6: hash table 크기와 같은 3개의 bucket으로 구성되어 있으며, 7개의 word가 입력되었음

(h) line 7: hash value = (sum of ascii codes) % (hash map size) = 0 인 word들과 입력파일에서의 빈도수.

(i) line 8: hash value 1 인 것들과 입력파일의 빈도수

(j) line 9: hash value 2 인 것들과 입력파일의 빈도수. 해당 단어가 없어서 출력되지 않음.

(k) line 12: hash map 의 단어들을 알파벳 순으로 출력함(hint: merge the buckets)

(l) lines 14-48: 다른 hash map size 와 다른 파일들에 대한 예. `example4-1.log`에 더 많은 예제가 있다.

```
1  $ ./hw4-1.exe 3 input/i31.txt output/i31_4-1_h3.txt
2  0.00026 seconds
3  324 bytes ( 46.286 x 7 )
4
5  $ cat output/i31_4-1_h3.txt # cat: display the text file
6  hashtable 3 buckets 7 words
7  bucket 0 (amet 3) (diam 6) (ipsum 5) (lorem 5) (sit 5)
8  bucket 1 (dolor 4) (sed 3)
9  bucket 2
10
11 alphabet-sorted 7 words
12 (amet 3) (diam 6) (dolor 4) (ipsum 5) (lorem 5) (sed 3) (sit 5)
13
14 $ ./hw4-1.exe 5 input/i31.txt output/i31_4-1_h5.txt
15 0.00034 seconds
16 356 bytes ( 50.857 x 7 )
17
18 $ cat output/i31_4-1_h5.txt # cat: display the text file
19 hashtable 5 buckets 7 words
20 bucket 0
```

```
bucket 1 (diam 6) (sed 3) (sit 5)
bucket 2
bucket 3 (amet 3) (ipsum 5) (lorem 5)
bucket 4 (dolor 4)

alphabet-sorted 7 words
(amet 3) (diam 6) (dolor 4) (ipsum 5) (lorem 5) (sed 3) (sit 5)

$ ./hw4-1.exe 11 input/imagine.txt output/imagine_4-1_h11.txt
0.00016 seconds
2667 bytes ( 39.806 x 67 )

$ cat output/imagine_4-1_h11.txt
hashtable 11 buckets 67 words
bucket 0 (hunger 1) (isn 1) (nothing 1) (sharing 1) (will 2)
bucket 1 (be 1) (but 2) (it 2) (no 6) (us 4)
bucket 2 (do 1) (greed 1) (sky 1) (the 8) (world 3)
bucket 3 (ah 1) (as 2) (hell 1) (join 2) (live 1) (one 4) (say 2)
bucket 4 (heaven 1) (imagine 6) (of 1) (peace 1) (possessions 1)
bucket 5 (all 4) (easy 1) (need 1) (or 2) (s 3)
bucket 6 (i 7) (in 1) (someday 2) (t 1) (today 1) (wonder 1)
bucket 7 (brotherhood 1) (livin 2) (ll 2) (not 2) (people 3) (to 2)
bucket 8 (above 1) (for 3) (hard 1) (man 1) (may 2) (there 2) (too 1) (you 8)
bucket 9 (a 3) (below 1) (can 1) (countries 1) (die 1) (if 2) (life 1)
bucket 10 (and 3) (dreamer 2) (hope 2) (kill 1) (m 4) (only 3) (religion 1) (try 1)

alphabet-sorted 67 words
(a 3) (above 1) (ah 1) (all 4) (and 3) (as 2) (be 1) (below 1) (brotherhood 1) (but 2) (can
     1) (countries 1) (die 1) (do 1) (dreamer 2) (easy 1) (for 3) (greed 1) (hard 1) (
    heaven 1) (hell 1) (hope 2) (hunger 1) (i 7) (if 2) (imagine 6) (in 1) (isn 1) (it 2)
    (join 2) (kill 1) (life 1) (live 1) (livin 2) (ll 2) (m 4) (man 1) (may 2) (need 1) (
    no 6) (not 2) (nothing 1) (of 1) (one 4) (only 3) (or 2) (peace 1) (people 3) (
    possessions 1) (religion 1) (s 3) (say 2) (sharing 1) (sky 1) (someday 2) (t 1) (the
    8) (there 2) (to 2) (today 1) (too 1) (try 1) (us 4) (will 2) (wonder 1) (world 3) (
    you 8)
```

template 코드 및 요구조건: `template_hw4-1.c`, `example4-1.log`, `input.zip`, `output.zip`

A. 활용가능한 구조체 `struct WORDNODE`, `struct WORDHASHTABLE`과 이 구조체들을 할당/반환할 수 있는 `create_wnode()`, `free_wnodelist_recursive()`, `create_word_hashtable()`, `free_hashtable()` 함수들이 주어진다. 요구조건을 만족하면 수정은 허용한다.

```
struct WORDNODE {
  char *word;  // content
  int count;   // number of appearances
  struct WORDNODE *next;
};

struct WORDHASHTABLE {
  struct WORDNODE **wnode;   // hash table, array of word node lists
  int size;    // size of the hash table
  int num_words;      // count unique number of words, same as the length of the list
};
```

B. "FILL" 로 표시가 되어 있는 영역에 코드 구현, 추가 함수 작성은 허용됨(해야함)

C. hash map size (bucket의 수)는 입력으로 주어진 값을 사용해야함

D. Hash map (table) 의 bucket 은 항상 정렬되어 있어야 한다. 즉, 하나의 word 를 삽입할 때마다 정렬된 linked list 를 유지한다.

E. 전체 sorted list 는 유지할 필요없고 파일 출력만 되면 된다.

F. 정확한 메모리 사용량을 측정하기 위해서는 배열을 사용할 경우에는 **malloc_c()** 함수를 사용하고, 문자열을 복사하여 사용할 경우 **strdup** 대신 **strdup_c** 함수를 사용한다. 사용예제는 `create_wnode(), create_word_hashtable()` 에 있다.

malloc_c(size_t size) 전역 변수 used_memory 에 size 를 누적하고 malloc을 호출

strcup_c(const char *s) 전역 변수 used_memory 에 s의 길이(strlen(s)+1)를 누적하고 strdup 호출

NOT ALLOWED 1 이 두 함수만을 이용하여 배열 사용이 허용됨. 예외 — 단어를 scanf로 읽어들이기 위한 main 함수의 buffer[256]. 배열이 아닌 일반 변수는 사용량에서 무시한다.

NOT ALLOWED 2 **malloc**, **strdup** 직접 사용하거나 **calloc, realloc** 등 다른 메모리 할당 함수는 사용하지 않는다(감점요인).

NOT ALLOWED 3 **memcpy, memccpy, memmove, wmemmove** 등의 함수는 사용하지 않는다.(cheating으로 간주되어 시간 점수 0점이 부여될 수 있음) 이 함수들은 시스템 HW call 등을 이용하여 매우 효율적으로 구현되어 있으며 UNIT OPERATION TIME 가정에 위배되기 때문에 제대로 평가를 할 수 없다.

제출물: `hw4-1.c` 오류 없이 컴파일 가능, 위의 실행예에 따라 실행가능해야 하며, 위의 예제의 형식에 맞는 출력 파일을 생성해야 한다.

채점기준 및 평가방법: Homework 4-1의 총점은 100점이며, 전체 성적에 8% 반영됨

10점 제출 기본점수

10점 학번/이름 주석문에 정확히 입력

40점 컴파일 오류 없고 실행이 정확한지 평가

- 예제로 주어진 입력과 다른 입력들에 대하여 정확한지 평가. 매우 큰 파일로 평가함. 각 예제별로 정답과 완전히 동일해야 한다 (한 글자도 틀리면 그 입력에 대하여 0점. line break 차이는 허용함. 출력들은 모두 공란 하나 ' '로 구분됨)
- 컴파일 오류시 실행불가로 0점이 주어짐
- 답이 맞더라도 알고리즘을 제대로 구현하지 않으면 해당과제를 하지 않은 것으로 판단하여 실행점수 0점

10점 실행시간 점수

- 10/5/0 3단계
- 10: 다른 학생들에 비해 월등하게 빠름
- 5: 거의 비슷함
- 0: 너무 오래 걸림

주관적이지만 알고리즘을 제대로 구현하였을 경우, 즉 대부분의 학생들이 5점을 받도록 분포를 보고 결정하겠습니다.

10점 메모리 사용량 점수

- 10/5/0 3단계
- 10: 다른 학생들에 비해 월등하게 사용량이 적음
- 5: 다른 학생들과 거의 비슷함
- 0: 너무 오래 걸림

마찬가지로 알고리즘을 제대로 구현하였을 경우, 즉 대부분의 학생들이 5점을 받도록 분포를 보고 결정하겠습니다. 출력된 메모리 사용량은 TA가 코드를 보고 누락된 것이 없는지 검증한다. 필요시 코드를 수정하거나, 수작업으로 값을 업데이트할 수 있으며 작성한 코드가 너무 복잡하여 메모리 사용량을 알 수 없을 경우 '판독불가'로 평가하고 점수를 부여하지 않는다.

20점 코드에 대한 평가(조건들 만족)

COPY 적발시 전체 0점. COPIED/BEING COPIED 모두 해당됨.

2 Homework 4-2

입력파일예: hw4-1과 동일

할것들: 주어진 텍스트 파일에서 단어를 하나씩 읽어서 chaining 을 이용한 hash table 에 저장한다. 중복되는 단어는 몇 개가 입력되었는지 그 수를 세어 저장(빈도수)하고, 각 bucket의 list는 빈도수의 내림차순, 그리고 단어의 alphabetic 오름차순으로 유지한다. 모든 단어를 읽고 난 후에 생성된 hash table 을 텍스트 파일에 출력하고, 전체 단어를 정렬하여 출력하고, **정렬순서로 입력파일을 encoding 하여 두번째 텍스트 파일에 출력한다.**

실행예 및 자세한 설명: 채점에 사용되는 command line argument 는 hash map size/입력파일/출력파일이다.

(a) line 1, hash map size (`argv[1]`): positive integer. 본 예제에서는 3

(b) line 1, 입력파일(`argv[2]`): 위에 예제로 보임

(c) line 1, 출력파일1(`argv[3]`): lines 6-18

(d) line 1, 출력파일2(`argv[4]`): line 20

(e) line 2, 화면출력줄1: **0.00019 seconds** 측정된 소요시간

(f) line 2, 화면출력줄2: **380 bytes (54.286 x 7)** 측정된 사용 메모리 동적 메모리로 사용된 메모리의 byte 수와 입력크기(7단어)의 배수로 표현됨

(g) line 5: '`cat`' 명령은 텍스트 파일을 화면에 출력함. 즉 '`output/i31_4-2_h3.txt`' 파일을 보임

(h) line 6: hash table 크기와 같은 3개의 bucket으로 구성되어 있으며, 7개의 word가 입력되었음

(i) line 7: hash value = (sum of ascii codes) % (hash map size) = 0 인 word들과 입력파일에서의 빈도수, **그리고 초기 code 값이 출력된다.** bucket 내의 단어들은 먼저 빈도수로 내림차순정렬되며 (즉 diam 의 6번이라 가장 먼저 나옴. ipsum/lorem/sit 은 같은 5번으로 알파벳순서) **아직 전체 정렬을 하지 않았기 때문에 code가 지정되지 않았음.** 따라서 초기값인 -1이 모두 출력됨

(j) line 8: hash value 1 인 것들과 입력파일의 빈도수

(k) line 9: hash value 2 인 것들과 입력파일의 빈도수. 해당 단어가 없어서 출력되지 않음.

(l) line 12: hash map 의 단어들을 **빈도수-내림차순/알파벳-오름차순**으로 출력함. 즉 amet이 알파벳 순서로는 가장 빠르지만 빈도수가 더 큰 diam, ipsum 등이 먼저 출력됨

(m) lines 14-17: code 값을 전체 sorting list의 순서로 업데이트하고 hash map을 다시 출력함. 추후 search를 위해서 hash map은 유지하고 있어야 한다.

(n) line 19: 두번째 출력파일인 '`encoded/i31_4-2_h3.txt`' 파일의 내용을 보임

(o) line 20: 입력파일의 단어들을 code로 mapping하여 파일을 생성한다. 즉 sit$\rightarrow 3$, lorem$\rightarrow 2$, dolor$\rightarrow 4$, diam$\rightarrow 0$, ...

(p) lines 22-57: 다른 hash map size 와 다른 파일들에 대한 예. `example4-2.log`에 더 많은 예제가 있다.

```
1  $ ./hw4-2.exe 3 input/i31.txt output/i31_4-2_h3.txt encoded/i31_4-2_h3.txt
2  0.00019 seconds
3  380 bytes ( 54.286 x 7 )
4
5  $ cat output/i31_4-2_h3.txt
6  hashtable 3 buckets 7 words
7  bucket 0 (diam 6 -1) (ipsum 5 -1) (lorem 5 -1) (sit 5 -1) (amet 3 -1)
8  bucket 1 (dolor 4 -1) (sed 3 -1)
9  bucket 2
10
11 count-alphabet-sorted 7 words
12 (diam 6 0) (ipsum 5 1) (lorem 5 2) (sit 5 3) (dolor 4 4) (amet 3 5) (sed 3 6)
13
14 hashtable 3 buckets 7 words
15 bucket 0 (diam 6 0) (ipsum 5 1) (lorem 5 2) (sit 5 3) (amet 3 5)
```

```
bucket 1 (dolor 4 4) (sed 3 6)
bucket 2

$ cat encoded/i31_4-2_h3.txt
3 2 4 0 0 2 6 4 0 3 5 0 0 2 1 2 1 5 3 2 6 1 3 1 5 4 4 6 3 1 0

$ ./hw4-2.exe 11 input/imagine.txt output/imagine_4-2_h11.txt encoded/imagine_4-2_h11.txt
0.00020 seconds
3203 bytes ( 47.806 x 67 )

$ cat output/imagine_4-2_h11.txt
hashtable 11 buckets 67 words
bucket 0 (will 2 -1) (hunger 1 -1) (isn 1 -1) (nothing 1 -1) (sharing 1 -1)
bucket 1 (no 6 -1) (us 4 -1) (but 2 -1) (it 2 -1) (be 1 -1)
bucket 2 (the 8 -1) (world 3 -1) (do 1 -1) (greed 1 -1) (sky 1 -1)
bucket 3 (one 4 -1) (as 2 -1) (join 2 -1) (say 2 -1) (ah 1 -1) (hell 1 -1) (live 1 -1)
bucket 4 (imagine 6 -1) (heaven 1 -1) (of 1 -1) (peace 1 -1) (possessions 1 -1)
bucket 5 (all 4 -1) (s 3 -1) (or 2 -1) (easy 1 -1) (need 1 -1)
bucket 6 (i 7 -1) (someday 2 -1) (in 1 -1) (t 1 -1) (today 1 -1) (wonder 1 -1)
bucket 7 (people 3 -1) (livin 2 -1) (ll 2 -1) (not 2 -1) (to 2 -1) (brotherhood 1 -1)
bucket 8 (you 8 -1) (for 3 -1) (may 2 -1) (there 2 -1) (above 1 -1) (hard 1 -1) (man 1 -1)
    (too 1 -1)
bucket 9 (a 3 -1) (if 2 -1) (below 1 -1) (can 1 -1) (countries 1 -1) (die 1 -1) (life 1 -1)
bucket 10 (m 4 -1) (and 3 -1) (only 3 -1) (dreamer 2 -1) (hope 2 -1) (kill 1 -1) (religion
    1 -1) (try 1 -1)

count-alphabet-sorted 67 words
(the 8 0) (you 8 1) (i 7 2) (imagine 6 3) (no 6 4) (all 4 5) (m 4 6) (one 4 7) (us 4 8) (a
    3 9) (and 3 10) (for 3 11) (only 3 12) (people 3 13) (s 3 14) (world 3 15) (as 2 16) (
    but 2 17) (dreamer 2 18) (hope 2 19) (if 2 20) (it 2 21) (join 2 22) (livin 2 23) (ll
    2 24) (may 2 25) (not 2 26) (or 2 27) (say 2 28) (someday 2 29) (there 2 30) (to 2 31)
     (will 2 32) (above 1 33) (ah 1 34) (be 1 35) (below 1 36) (brotherhood 1 37) (can 1
    38) (countries 1 39) (die 1 40) (do 1 41) (easy 1 42) (greed 1 43) (hard 1 44) (heaven
     1 45) (hell 1 46) (hunger 1 47) (in 1 48) (isn 1 49) (kill 1 50) (life 1 51) (live 1
    52) (man 1 53) (need 1 54) (nothing 1 55) (of 1 56) (peace 1 57) (possessions 1 58) (
    religion 1 59) (sharing 1 60) (sky 1 61) (t 1 62) (today 1 63) (too 1 64) (try 1 65) (
    wonder 1 66)

hashtable 11 buckets 67 words
bucket 0 (will 2 32) (hunger 1 47) (isn 1 49) (nothing 1 55) (sharing 1 60)
bucket 1 (no 6 4) (us 4 8) (but 2 17) (it 2 21) (be 1 35)
bucket 2 (the 8 0) (world 3 15) (do 1 41) (greed 1 43) (sky 1 61)
bucket 3 (one 4 7) (as 2 16) (join 2 22) (say 2 28) (ah 1 34) (hell 1 46) (live 1 52)
bucket 4 (imagine 6 3) (heaven 1 45) (of 1 56) (peace 1 57) (possessions 1 58)
bucket 5 (all 4 5) (s 3 14) (or 2 27) (easy 1 42) (need 1 54)
bucket 6 (i 7 2) (someday 2 29) (in 1 48) (t 1 62) (today 1 63) (wonder 1 66)
bucket 7 (people 3 13) (livin 2 23) (ll 2 24) (not 2 26) (to 2 31) (brotherhood 1 37)
bucket 8 (you 8 1) (for 3 11) (may 2 25) (there 2 30) (above 1 33) (hard 1 44) (man 1 53) (
    too 1 64)
bucket 9 (a 3 9) (if 2 20) (below 1 36) (can 1 38) (countries 1 39) (die 1 40) (life 1 51)
bucket 10 (m 4 6) (and 3 10) (only 3 12) (dreamer 2 18) (hope 2 19) (kill 1 50) (religion 1
     59) (try 1 65)

$ cat encoded/imagine_4-2_h11.txt
3 30 14 4 45 21 14 42 20 1 65 4 46 36 8 33 8 12 61 3 5 0 13 23 11 63 34 3 30 14 4 39 21 49
    62 44 31 41 55 31 50 27 40 11 10 4 59 64 3 5 0 13 23 51 48 57 1 1 25 28 2 6 9 18 17 2
    6 26 0 12 7 2 19 29 1 24 22 8 10 0 15 32 35 16 7 3 4 58 2 66 20 1 38 4 54 11 43 27 47
    9 37 56 53 3 5 0 13 60 5 0 15 1 1 25 28 2 6 9 18 17 2 6 26 0 12 7 2 19 29 1 24 22 8 10
     0 15 32 52 16 7
```

template 코드 및 요구조건: `template_hw4-2.c`, `example4-2.log`, `input.zip`, `output.zip`. hw4-1 과의 차이만을 설명함

A. `struct WORDNODE` 에 `code` 변수 추가됨

B. 하나의 word 를 삽입할 때마다 정렬된 linked list 를 유지한다. 정렬은 count의 내림차순으로 먼저 정렬하고, 그 이후 alphabet의 오름차순을 유지한다. (note: 이미 hash map 에 있는 단어를 삽입하면 count 값이 변하고, 리스트의 원소 위치가 변할 수 있다.)

C. 전체 sorted list 는 유지할 필요없고 파일 출력만 되면 된다. code값은 전체 sorting 이후 업데이트 한다.

D. encoding 파일을 반드시 생성해야 한다.

제출물: `hw4-2.c` 오류 없이 컴파일 가능, 위의 실행예에 따라 실행가능해야 하며, 위의 예제의 형식에 맞는 출력 파일을 생성해야 한다.

채점기준 및 평가방법: Homework 4-2의 총점은 100점이며, 전체 성적에 7% 반영됨. 채점기준은 hw4-1 과 동일

3 제출형식 및 방법

제출할 코드 `hw4-1.c`, `hw4-2.c` 만을 제출해야 함. 필요없는 파일 제출시 10점까지 감점 있음

제출방법 위의 2개의 파일을 묶어서 `hw4.zip` 을 만들고, `lms.knu.ac.kr` 에 업로드한다. LMS는 제출된 파일의 이름을 복잡하게 바꾸기 때문에 개별파일을 제출하면 채점이 매우 어렵다.

Due 6/3 목요일 11:59 LMS time

Late submission 6/4 금요일 09:59 LMS time, 시간당 10점 감점