1. 다음과 같은 조건을 만족시키는 다중 모듈 프로그램을 C로 작성하시오.
   - main() 함수 포함하는 한 개의 프로그램
     1. 1개의 문자열 A와 입력 파일(infile)에서 한 줄인 B를 입력 받음.
     2. 문자열 A의 크기가 1023를 넘을 경우 "Too long"을 출력하고 종료함
     3. 파일(infile)의 끝까지 A와 동일한 부분이 있는지 찾는 함수를 호출
     4. 동일한 부분이 있으면 해당 줄(line)과 줄번호를 출력함. 파일에 A와 동일한 부분이 하나도 없으면 "No pattern"을 출력함
   - 문자열과 관련한 함수를 포함하는 프로그램
     1. 문자열의 길이를 반환하는 함수가 존재함
     2. 문자열 내의 모든 문자를 소문자로 변환하는 함수를 작성
   - A에서 B와 같은 패턴이 있는지 확인하고 TRUE/FALSE를 반환하는 프로그램

■ Due Date: 6월 15일(수)

2. 앞 페이지의 기능을 수행하는 프로그램을 linux_hw7 디렉터리 내에 아래와 같이 2개의 헤더 2개의 C파일로 작성하고, 컴파일 하여 올바르게 실행시키기 위한 Makefile을 작성하시오.

- 즉, 각각의 프로그램을 limit.h, str.h, str.c, sgrep.c라는 이름으로 저장함.

3. 다음과 같은 소스 파일과 결과로 출력된 캡쳐 파일 하나의 한글/Word 파일(hw7.hwp 또는 hw7.doc)로 만들어 e-class 과제방에 제출하시오.

- linux_hw7 디렉터리에서 (date; ls -lf)를 실행한 결과
- sgrep.c 프로그램 소스 코드와 Makefile 내용
- 고정된 파일(`infile`)과 입력받은 문자열에 대해 sgrep을 실행한 결과를 캡쳐한 파일

  쉘에서 실행 명령: sgrep computer

- limits.h 파일

```
#ifndef _LIMITS_H_
#define _LIMITS_H_

#define MAX_STR_LEN 1023

#endif /* _LIMITS_H_ */
```

- str.h

```
#ifndef _STR_H_
#define _STR_H_
#include <limits.h> /* for MAX_STR_LEN */
#include <unistd.h> /* for typedef of size_t */

size_t StrGetLength(const char* pcSrc);
char *StrToLower(char *str);

#endif /* _STR_H_ */
```

- str.c 파일

```
#include <assert.h> /* to use assert() */
#include <stdio.h>
#include <string.h>
#include "str.h"
#include <ctype.h>
```

```
/* Your task is:
   Rewrite the body of "Part 1" functions - remove the current body that simply calls the corresponding C
   standard library function.
*/
/*--------------------------------------------------------------------------*/
size_t StrGetLength(const char* pcSrc)
{
  const char *pcEnd;
  assert(pcSrc); /* NULL address, 0, and FALSE are identical. */
  pcEnd = pcSrc;

  while (*pcEnd) /* null character and FALSE are identical. */
    pcEnd++;

  return (size_t)(pcEnd - pcSrc);
}
/*--------------------------------------------------------------------------*/
char *StrToLower(char *str)
{
  char *str_clone;
  str_clone = (char *)malloc(1024);

  int count=0;
  /* TODO: fill this function */
  /* Part 1 */
}
```

- sgrep.c 파일

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h> /* for skeleton code */
#include <unistd.h> /* for getopt */
#include "str.h"

#define FALSE 0
#define TRUE  1

/*
 * Fill out your own functions here (If you need)
 */

/*--------------------------------------------------------------------*/
/* PrintUsage()
   print out the usage of the Simple Grep Program                     */
/*--------------------------------------------------------------------*/
void PrintUsage(const char* argv0)
{
  const static char *fmt =
                "Simple match (match) Usage:₩n"
                "%s pattern ₩n";

  printf(fmt, argv0);
}
```

```
/*-------------------------------------------------------------*/
/* SearchPattern()
   Your task:
   1. Do argument and input string validation
   - String or file argument length is no more than 1023
   - If you encounter an input argument that's too long, print out "Error: pattern is too long"

   2. Read the each line from input file(infile)
   - If you encounter a line larger than 1023 bytes, print out "Error: input line is too long"
   - Error message should be printed out to standard error (stderr)

   3. Check & print out the line contains a given string (search-string)

   Tips:
   - fgets() is an useful function to read characters from file. Note
   that the fget() reads until newline or the end-of-file is reached.
   - fprintf(sderr, ...) should be useful for printing out error
   message to standard error

   NOTE: If there is any problem, return FALSE; if not, return TRUE  */
```

```
/*---------------------------------------------------------------------*/
int SearchPattern(const char *pattern)
{
  char buf[MAX_STR_LEN + 2];
  FILE *fp;
  if( (fp = fopen("infile", "r")) == NULL) {
                fprintf(stderr, "Error: file open error₩n);
                return(EXIT_FAILURE);
  }
  /*
   *  TODO: check if pattern is too long and there exists
   */
 return TRUE;
}

/*---------------------------------------------------------------------*/
int main(const int argc, const char *argv[])
{
  /* Do argument check and parsing */
  if (argc < 1) {
                fprintf(stderr, "Error: argument parsing error₩n");
                PrintUsage(argv[0]);
                return (EXIT_FAILURE);
  }
  return SearchPattern(argv[1]) ? EXIT_SUCCESS:EXIT_FAILURE;
}
```