Exercise 1(grep): create a project, named as **grep**, and add 3 source files, main.cpp, getline.cpp and strindex.cpp as Figure 1 and Figure 2. Moreover use 記事本 to edit a text file article.txt which contains a part of an article like Figure 3.

```
#include <stdio.h>
#define  MAXLINE    1000  /* maximum input line length */

/* declaration of function prototype in order to do type checking */
int getline( char s[], int lim ) ;

int strindex( char s[], char t[] ) ;

char pattern[] = "ould" ; /* pattern to search for */

int main( int argc, char *argv[]  )
{
    char line[MAXLINE] ;
    int  found = 0 ;

    while( 0 < getline(line, MAXLINE) ){
        if (  0 <= strindex(line, pattern) ){
            printf("%s\n", line );
            found++ ;
        }// end if
    }// end while
    return found ;
}
```

Figure 1: source code in main.cpp

```
#include <stdio.h>

/*  getline: get line into s, return length
 *
 *  input  -   lim : maximum size of character array s
 *  ouptut -   s : contains one line from stdin
 *
 *  return lengh of string s
 */

int getline( char s[], int lim )
{
    int c, i ;

    i = 0 ;
    while(  --lim > 0  &&  (c=getchar()) != EOF && c != '\n'  ){
        s[ i++ ] = c ;
    }
    s[i] = '\0' ;
    return i ;
}
```

```
int strindex( char s[], char t[] )
{
    int i, j, k ;


    for ( i = 0 ; '\0' != s[i] ; i++ ){

        for ( j=i, k=0 ; '\0' != t[k]  && s[j] == t[k] ; j++, k++) {
            ;
        } // for j

        if ( 0 < k && '\0' == t[k] ){ // match whole pattern in t
            return i ;
        }
    }// for i

    return -1 ; // don't match pattern in t
}
```

Figure 2: left panel is source code in getline.cpp and right panel is source code in strindex.cpp

```
Ah Love! could you and I with Fate conspire
To grasp this sorry Scheme of Things entire,
Would not we shatter it to bite -- and then
Re-mould it nearer to the Heart's Desire!
```

Figure 3: article.txt

(1) why we parameter “int lim” in function **getline**, what is the functionality of “int lim”? Can we pass this parameter any value?

(2) Why matching criterion is **0 < k && '\0' == t[k]** , can you use another matching criterion?

(3) Upload your project to Linux machine and use icpc to compile it, execute command **./a.out < article.txt** and **grep ould article.txt**

check their results

(4) Issue command **g++ -v main.cpp getline.cpp strindex.cpp** , check the compilation order of three source files, how about if we use command **icpc -v main.cpp getline.cpp strindex.cpp**

(5) In function getline, can we use

--lim > 0 && c != '\n' && (c=getchar()) != EOF

write a program to test this when feeding with article.txt

Exercise 2(CPP, macro substitution): In mathematical header file **math.h**, we don't have maximum or minimum operation, but these two operations can be achieved by using macro substitution.

```c
#include <stdio.h>

#define  MAX( A, B ) ( (A) > (B) ? (A) : (B) )

int main( int argc, char* argv[] )
{
    int x,y,z ;

    x = 3 ;
    y = 5 ;

    z = MAX( x, y ) ;
    x++ ;
    y++ ;

    printf("x = %d, y = %d, z = %d\n", x, y, z );

    return 0 ;
}
```

(a)

```c
#include <stdio.h>

#define  MAX( A, B ) ( (A) > (B) ? (A) : (B) )

int main( int argc, char* argv[] )
{
    int x,y,z ;

    x = 3 ;
    y = 5 ;

    z = MAX( x++, y++ ) ;

    printf("x = %d, y = %d, z = %d\n", x, y, z );

    return 0 ;
}
```

(b)

Figure 4: (a) take z be maximum of x and y, then increment x and y by 1. Code (b) combines maximum operation and increment operation into one statement.

(1) Suppose we want to do two operations, one is $z = \max(x, y)$ and then increment x, y by 1 respectively, like code in Figure 4(a). What is the result?

(2) Someone wants to combine these two operations into one in order to reduce number of lines of the program, the resulting code is like Figure 4(b). What is the result of (b), is it the same as result in (a)? Hint: use cpp to parse (b) first without compilation.

Exercise 3(CPP, macro substitution): In page 90 of textbook, the author says macro

#define square(x) x*x

is wrong, since $square(z+1) \neq (z+1)^2$, can you interpret this? What is correct version of square macro? Write program to test it.

Exercise 4(CPP, conditional inclusion): Sometimes we write a program complied in different platform (windows, linux). Since different platform may support different library, we need to tell compiler which library we want, this is why we use **conditional inclusion**. Consider simple test program in Figure 5, we use macro _WIN32 (or macro __WIN32__) to tell compiler which code we want, “printf("This is Win32 Application\n");” or “printf("This is Linux Application\n");”.

```
#include <stdio.h>

int main( int argc, char* argv[] )
{
    #if defined(_WIN32) || defined(__WIN32__)
            printf("This is Win32 Application\n");
    #else
            printf("This is Linux Application\n");
    #endif

    return 0 ;
}
```

Figure 5: macro _WIN32 is Microsoft-specific, it is always defined by Visual Studio, but not defined in configuration of gcc or icpc.

(1) Write this program stored in file **main.cpp** and then verify the result of program on Windows (through Visual Studio) and on Linux (compiled by gcc or icpc)

(2) Try compile the program by

g++ –D_WIN32=1 main.cpp

what is the result you obtain?

(3) What is the program after preprocessing stage? Hint: use program **cpp (C preprocessor)** to parse the program.

Exercise 5(CPP, causality 因果律): In project **grep** in **Exercise 1**, we add two header files, one is **getline.h** and the other is **strindex.h**. We code these two header files including each other, we have shown that Visual Studio report error “infinite recursion”.

getline.h

```
/* test for nested file inclusion */
#include "strindex.h"

/* prototype of function getline */
int getline( char s[], int lim ) ;
```

strindex.h

```
/* test for nested file inclusion */
#include "getline.h"

/* prototype of function strindex */
int strindex( char s[], char t[] ) ;
```

(1) Execute command “icpc main.cpp getline.cpp strindex.cpp” in Linux machine, what is the error message?

(2) Execute command “g++ main.cpp getline.cpp strindex.cpp” in Linux machine, what is the error message? Which compiler is better as far as you are concerned?

(3) We propose a solution to solve this infinite recursion by using **conditional inclusion** as Figure 6. Modify header files getline.h and strindex.h as Figure 6 and then execute command “cpp strindex.cpp”, what is the result? Compare this result with we talk about in power point.

getline.h

```
#ifndef  GETLINE_H

#define  GETLINE_H

/* test for nested file inclusion */
    #include "strindex.h"

/* prototype of function getline */
    int getline( char s[], int lim ) ;

#endif  // GETLINE_H
```

strindex.h

```
#ifndef  STRINDEX_H

    #define  STRINDEX_H

/* test for nested file inclusion */
    #include "getline.h"

/* prototype of function strindex */
    int strindex( char s[], char t[] ) ;

#endif  // STRINDEX_H
```

Figure 6: use conditional inclusion to solve infinite recursion

Exercise 6(atof): in page 71 of textbook, the author write a function called atof, which converts the string *s* to its double-precision floating-point equivalent. Now create a project named as **atof,** and add two source files into this project, one is **main.cpp (**Figure 7**)** and the other is **atof.cpp (**Figure 8**)**.

```
#include <stdio.h>
#include <math.h>

double atof( char s[] ) ;

int main(int argc, char* argv[] )
{
    char str[128] ;

    printf("atof(%s) = %25.16E\n", "314", atof("314") ) ;
    printf("atof(%s) = %25.16E\n", "3.14", atof("3.14") ) ;
    printf("atof(%s) = %25.16E\n", "3a.14", atof("3a.14") ) ;
    printf("atof(%s) = %25.16E\n", "3.1b4", atof("3.1b4") ) ;
    printf("atof(%s) = %25.16E\n", "3.14E-2", atof( "3.14E-2" ) ) ;

    sprintf( str, "%25.17f", atan(1.0) ) ;
    printf("atof(%s) = %25.16E\n", str , atof( str ) ) ;

    return 0 ;
}
```

Figure 7: source code in file main.cpp, function sprintf is described in page 245 of textbook.

```
#include <ctype.h>
/* atof: conver string s to double */
double atof( char s[] )
{
    double val, power ;
    int i, sign ;

    for( i=0 ; isspace(s[i]) ; i++) /* skip white space*/
        ;
    sign = ( '-' == s[i] )? -1 : 1 ;
    if ( '+' == s[i] || '-' == s[i] )
        i++ ;
    for(val = 0.0 ; isdigit(s[i]) ; i++ )
        val = 10.0 * val + ( s[i] - '0' ) ;
    if ( '.' == s[i] )
        i++ ;
    for( power = 1.0 ; isdigit(s[i]) ; i++){
        val = 10.0 * val + ( s[i] - '0' ) ;
        power *= 10.0 ;
    }
    return sign * val / power ;
}
```

Figure 8: source code in file atof.cpp, note that standard C library has function atof in stdlib.h

(1) Find scope and lifetime of each variable in **main.cpp** and **atof.cpp**
(2) When you execute the program, function atof cannot transform "3a.14", "3.1b4" and "3.14E-2" into correct form, why?
(3) Can you modify function **atof** such that it can work for "3.14E-2"?
(4) Standard C library provides function **atof** in stdlib.h, could you use this built-in library function to test all examples in main.cpp?

Exercise 7(operator precedence versus grammar): in chapter 3, we introduce grammar of expression, which is written in page 237~238 of textbook, in this chapter (chapter 4), we introduce operator precedence and associativity (see page 53 of textbook or search this in MSDN library), can you find relationship between grammar and operator precedence? Take some example to demonstrate your discovery.
Hint: operator precedence corresponds to order of definition of grammar and associativity relates to left / right recursion of grammar