Object Oriented Programming (Week 5)

2023

KWANGWOON UNIVERSITY

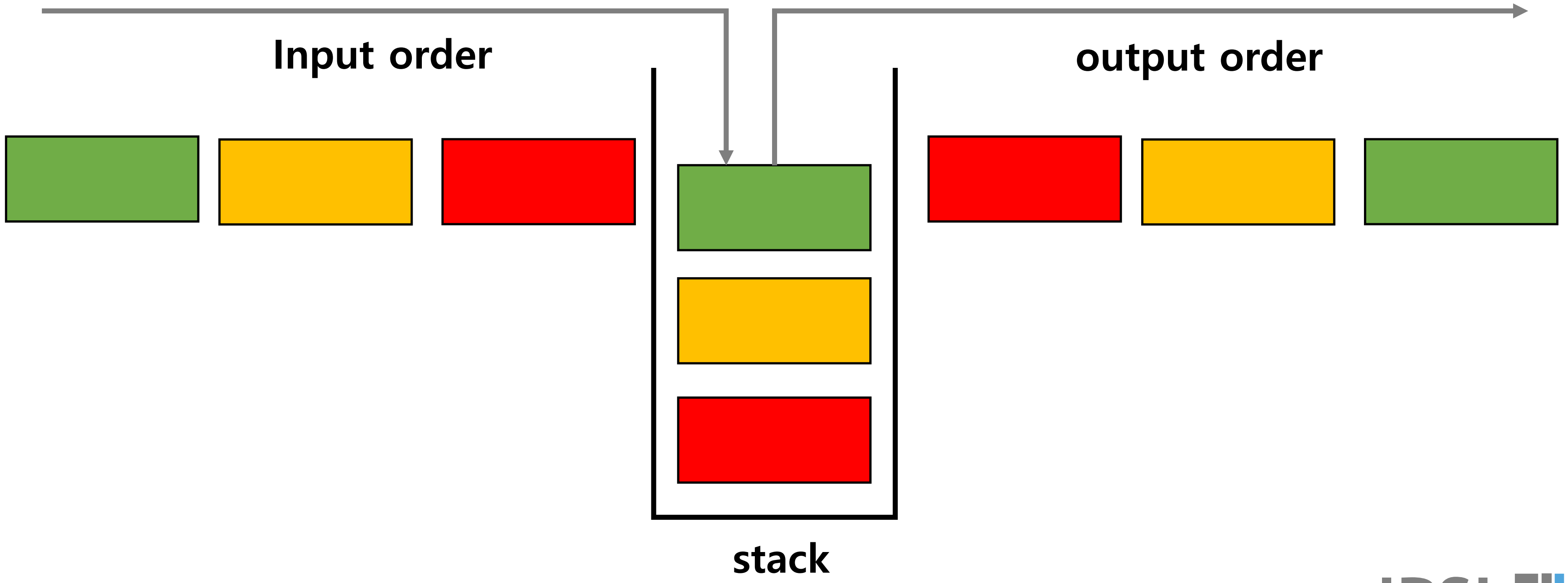
DEPT. OF COMPUTER ENGINEERING

Contents

ASSIGNMENT 2-1. 1

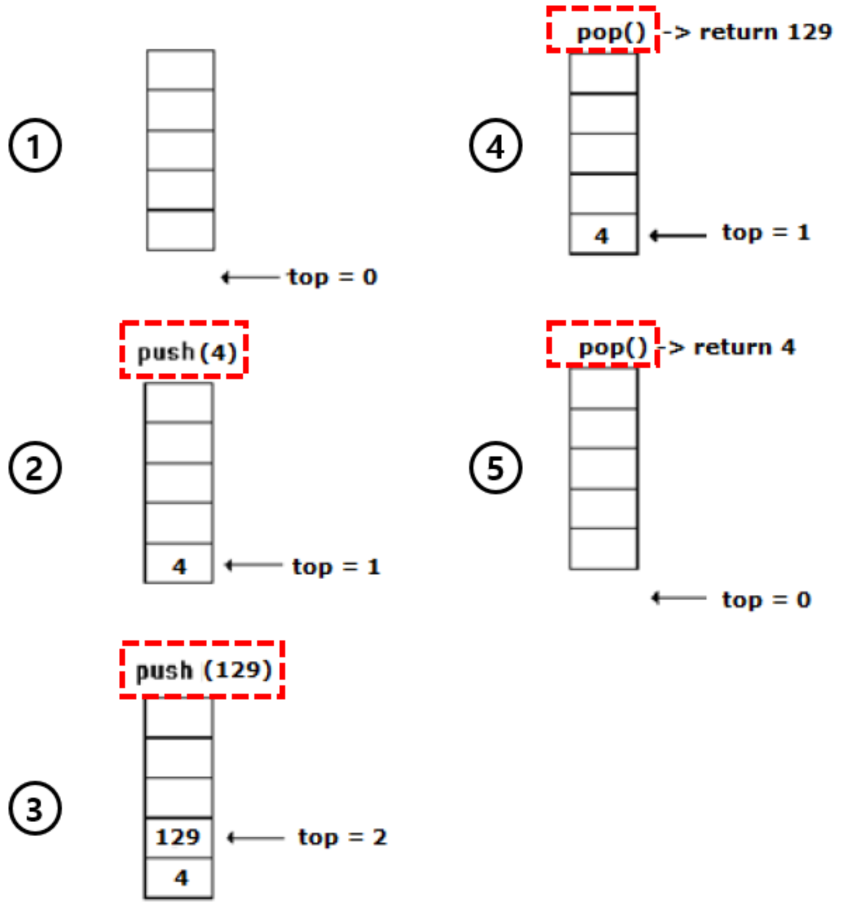
Assignment 2-1. 1

- (**Stack**) A stack is a linear data structure in computer science that stores a collection of elements, where elements are added or removed only at one end, called the "**top**."
- It follows the Last-In-First-Out (**LIFO**) principle, where the most recently added element is the first to be removed.



Assignment 2-1. 1

- To better understand the stack, please refer to the illustrative figure below.



Assignment 2-1. 1

- You are required to implement the following commands in the table to create a stack that operates based on given descriptions.

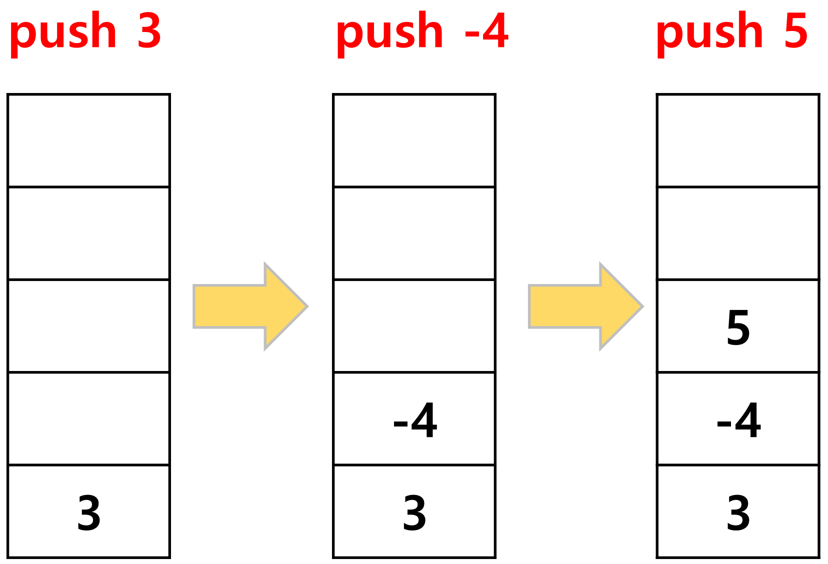
Command	Description
push [int number]	Add value to the top of the stack; no print
pop	Print the value at the top and remove the value from the stack
top	Print the value at the top
print	Print all values of the stack in a single line, separated by a white space, in the order from the bottom to the top
empty	Print 1 if the stack is empty, or print 0 if it is not
exit	Terminate program

Assignment 2-1. 1

Input	Output
push 3	
push -4	
push 5	
push 7	
push 8	
print	3 -4 5 7 8
top	8
pop	8
pop	7
pop	5
print	3 -4
empty	0
pop	-4
pop	3
empty	1
exit	

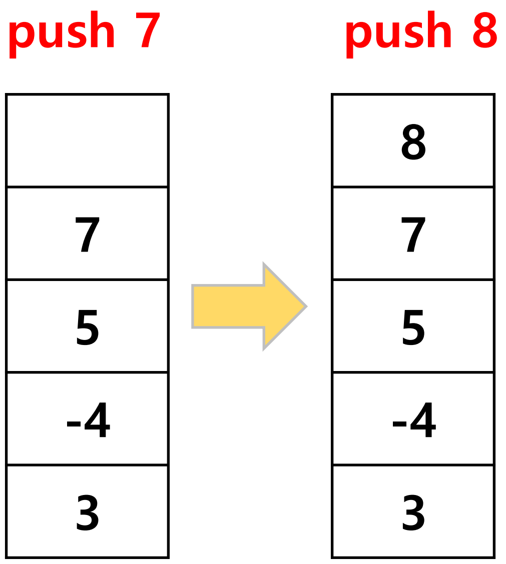
Assignment 2-1. 1

Input	Output
push 3	
push -4	
push 5	
push 7	
push 8	
print	3 -4 5 7 8
top	8
pop	8
pop	7
pop	5
print	3 -4
empty	0
pop	-4
pop	3
empty	1
exit	



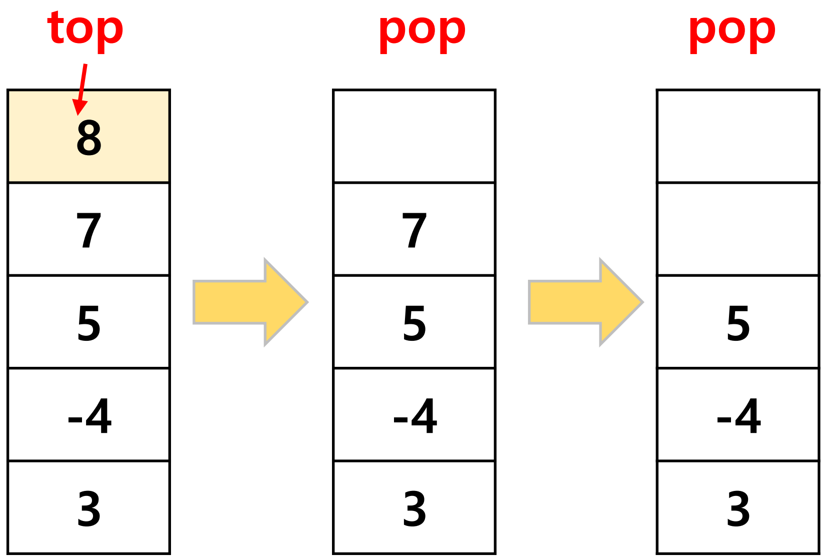
Assignment 2-1. 1

Input	Output
push 3	
push -4	
push 5	
push 7	
push 8	
print	3 -4 5 7 8
top	8
pop	8
pop	7
pop	5
print	3 -4
empty	0
pop	-4
pop	3
empty	1
exit	



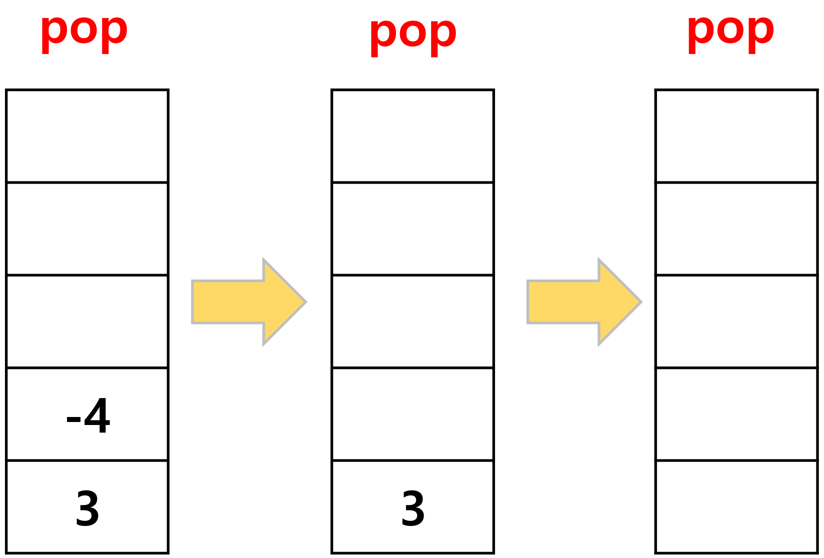
Assignment 2-1. 1

Input	Output
push 3	
push -4	
push 5	
push 7	
push 8	
print	3 -4 5 7 8
top	8
pop	8
pop	7
pop	5
print	3 -4
empty	0
pop	-4
pop	3
empty	1
exit	



Assignment 2-1. 1

Input	Output
push 3	
push -4	
push 5	
push 7	
push 8	
print	3 -4 5 7 8
top	8
pop	8
pop	7
pop	5
print	3 -4
empty	0
pop	-4
pop	3
empty	1
exit	



ASSIGNMENT 2-1. 2

Assignment 2-1. 2

- (Maze) Given a maze, **find the shortest path** from the starting point to the destination point and output the **total length** of the path as an integer.
- The first line contains the size of the maze in the format of "**number of rows**" and "**number of columns**," separated by a space.
  - The maze size is between **1 and 30** for both rows and columns.

Input	Output
4 5	6
10110	
00001	
10101	
10101	
1 2 4 4	

Assignment 2-1. 2

- Starting from the second line, the maze of **0's (path)** and **1's (wall)** is input without spaces.

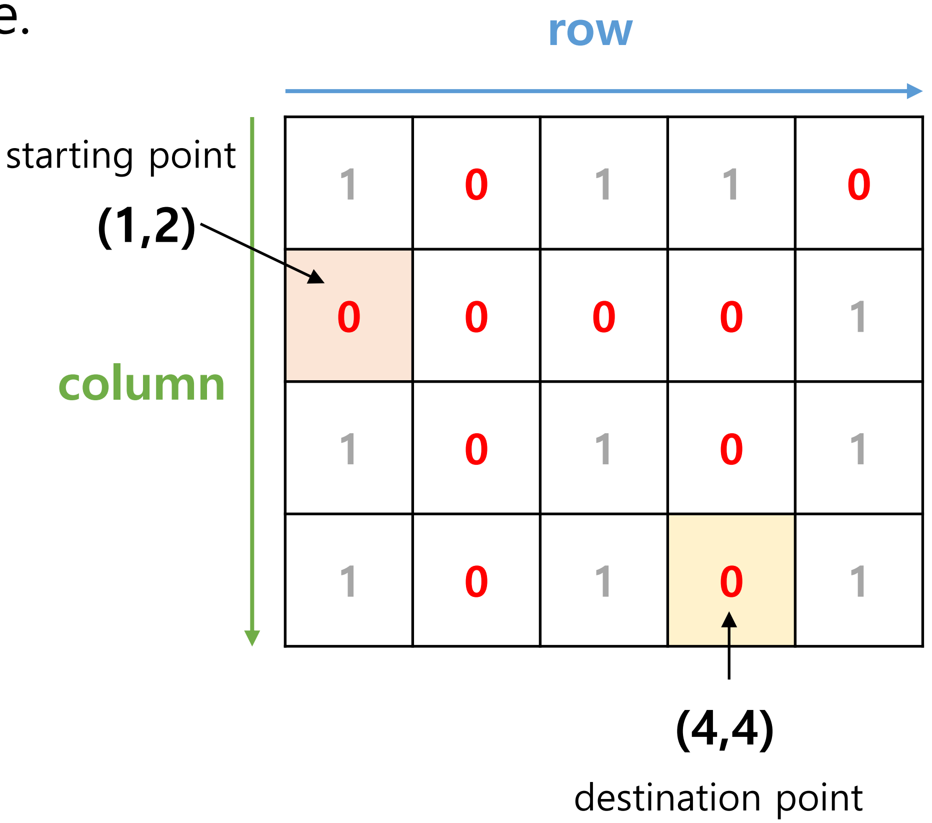
Input
4 5
10110
00001
10101
10101
1 2 4 4

1	0	1	1	0
0	0	0	0	1
1	0	1	0	1
1	0	1	0	1

Assignment 2-1. 2

- In the **last** line, the **starting point** and the **destination point** are given as the coordinates in the format of "row of the starting point," "column of the starting point," "row of the destination point," "column of the destination point," separated by a space.

Input
4 5
10110
00001
10101
10101
1 2 4 4

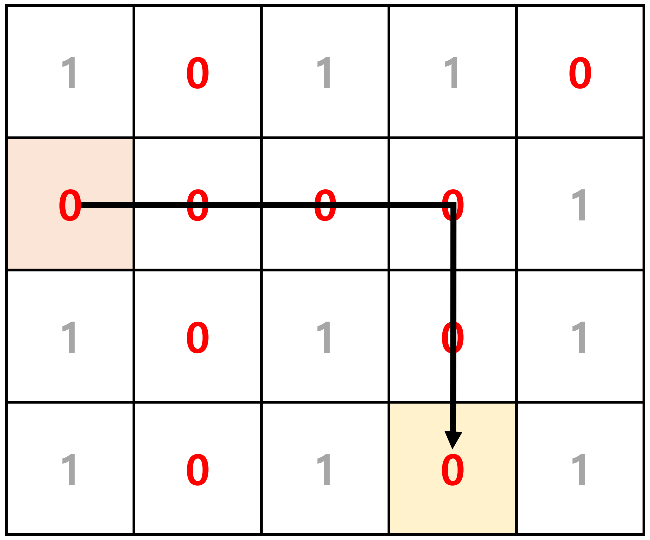


Assignment 2-1. 2

- In the maze, movement is only possible in the **four directions** of **east, west, south, and north**, and you cannot move to a place blocked by a wall.
- The path from the starting point to the destination point in the maze is given as one and only one case.
- The path **includes** both the starting and destination points.

Input	Output

4 5
10110
00001
10101
10101
1 2 4 4
	6



ASSIGNMENT 2-1. 3

Assignment 2-1. 3

- (String function) Write functions to manipulate **1D array** in oopstd namespace and compare them to the standard functions defined in standard headers (string, cstdlib).
- You can see the cplusplus documentation to understand the detailed behavior.
  - https://cplusplus.com/reference/

Assignment 2-1. 3

oopstd.h

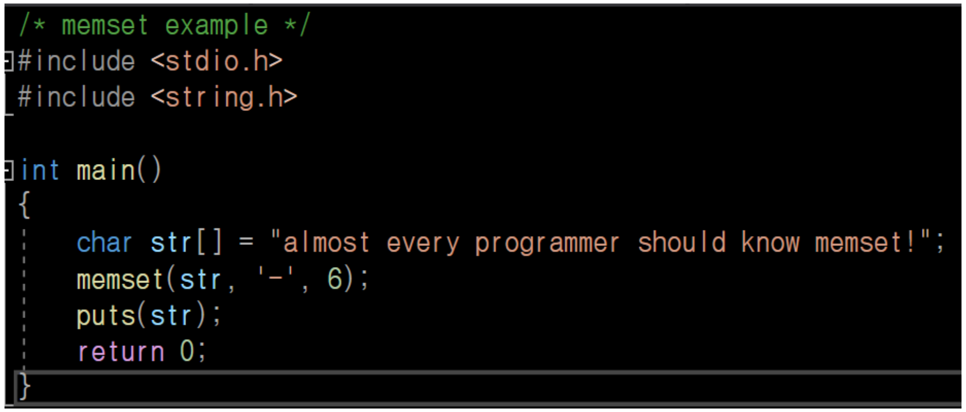
```
namespace oopstd {

void* memset(void* ptr, int value, size_t num);
void* memcpy(void* destination, const void* source, size_t num);

int strcmp(const char* str1, const char* str2);
int strncmp(const char* str1, const char* str2, size_t num);

char* strcpy(char* destination, const char* source);
char* strncpy(char* destination, const char* source, size_t num);

size_t strlen(const char* str);

int atoi(const char* str);
float atof(const char* str);

}
```

Assignment 2-1. 3

- **memset**
  - void * memset (void * ptr, int value, size_t num);
  - Sets the first num bytes of the block of memory pointed by ptr to the specified value (interpreted as an unsigned char)
  - return value
    - *ptr* is returned.

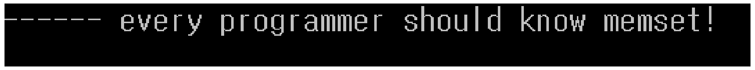
- **memcpy**
  - void * memcpy (void * destination, const void * source, size_t num);
  - Copies the values of num bytes from the location pointed to by source directly to the memory block pointed to by destination
  - return value
    - *destination* is returned.

Assignment 2-1. 3

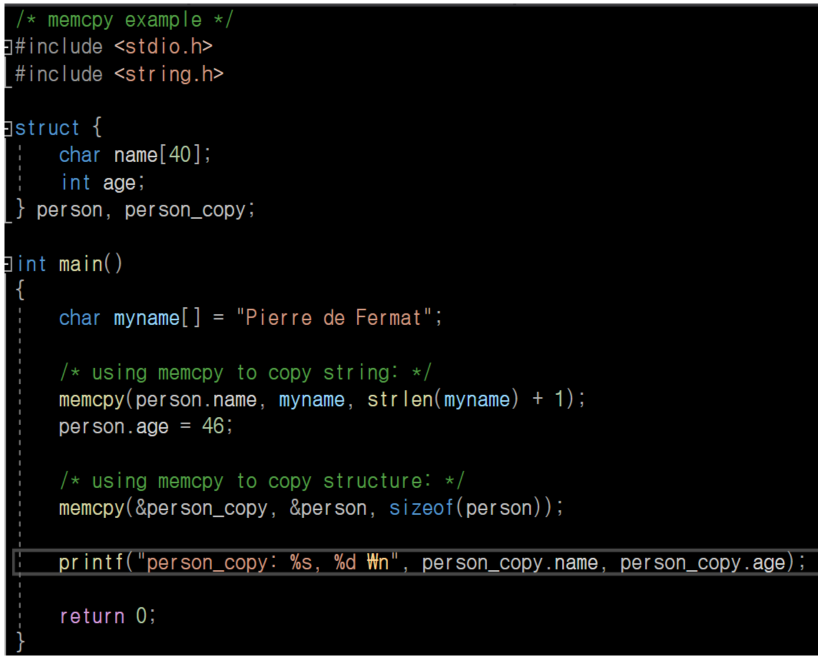


```
/* memset example */
#include <stdio.h>
#include <string.h>

int main()
{
    char str[] = "almost every programmer should know memset!";
    memset(str, '-', 6);
    puts(str);
    return 0;
}
```

출력

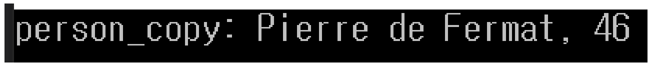


```
------ every programmer should know memset!
```



```
/* memcpy example */
#include <stdio.h>
#include <string.h>

struct {
    char name[40];
    int age;
} person, person_copy;

int main()
{
    char myname[] = "Pierre de Fermat";

    /* using memcpy to copy string: */
    memcpy(person.name, myname, strlen(myname) + 1);
    person.age = 46;

    /* using memcpy to copy structure: */
    memcpy(&person_copy, &person, sizeof(person));

    printf("person_copy: %s, %d ₩n", person_copy.name, person_copy.age);

    return 0;
}
```

출력



```
person_copy: Pierre de Fermat, 46
```

Assignment 2-1. 3

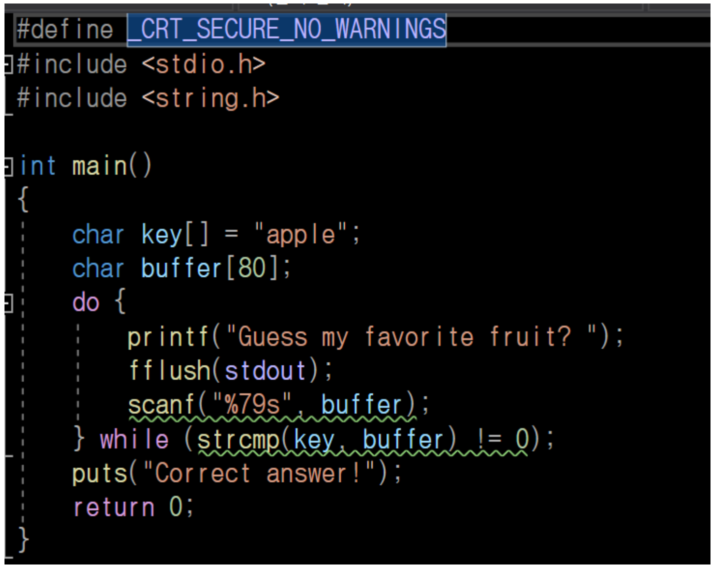
- **strcmp**
  - int strcmp (const char * str1, const char * str2);
  - Compares the C string *str1* to the C string *str2*
  - return value

return value	indicates
<0	the first character that does not match has a lower value in *ptr1* than in *ptr2*
0	the contents of both strings are equal
>0	the first character that does not match has a greater value in *ptr1* than in *ptr2*

- **strncmp**
  - int strncmp (const char * str1, const char * str2, size_t num);
  - Compares up to *num* characters of the C string *str1* to those of the C string *str2*
  - return value

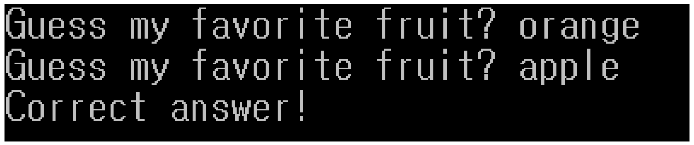
return value	indicates
<0	the first character that does not match has a lower value in *str1* than in *str2*
0	the contents of both strings are equal
>0	the first character that does not match has a greater value in *str1* than in *str2*

Assignment 2-1. 3

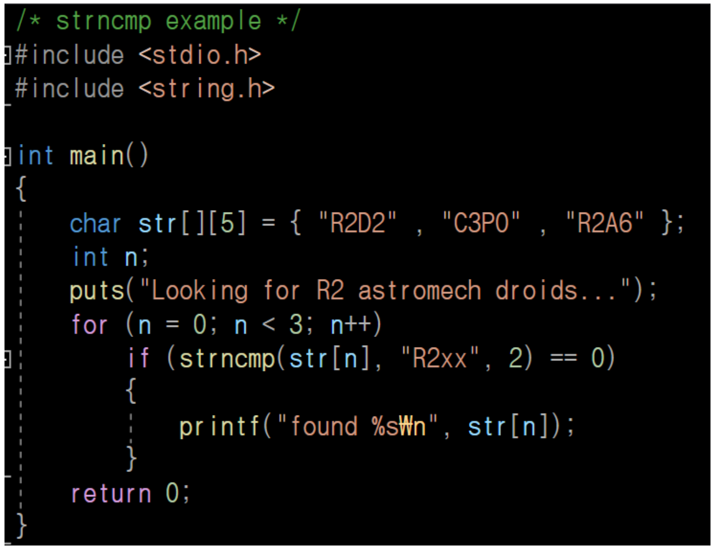


```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h>

int main()
{
    char key[] = "apple";
    char buffer[80];
    do {
        printf("Guess my favorite fruit? ");
        fflush(stdout);
        scanf("%79s", buffer);
    } while (strcmp(key, buffer) != 0);
    puts("Correct answer!");
    return 0;
}
```

출력

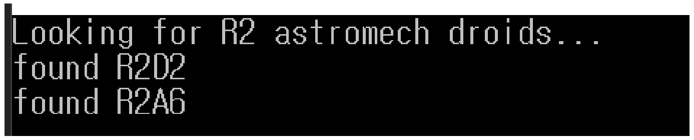


```
Guess my favorite fruit? orange
Guess my favorite fruit? apple
Correct answer!
```



```c
/* strncmp example */
#include <stdio.h>
#include <string.h>

int main()
{
    char str[][5] = { "R2D2" , "C3PO" , "R2A6" };
    int n;
    puts("Looking for R2 astromech droids...");
    for (n = 0; n < 3; n++)
        if (strncmp(str[n], "R2xx", 2) == 0)
        {
            printf("found %s₩n", str[n]);
        }
    return 0;
}
```

출력



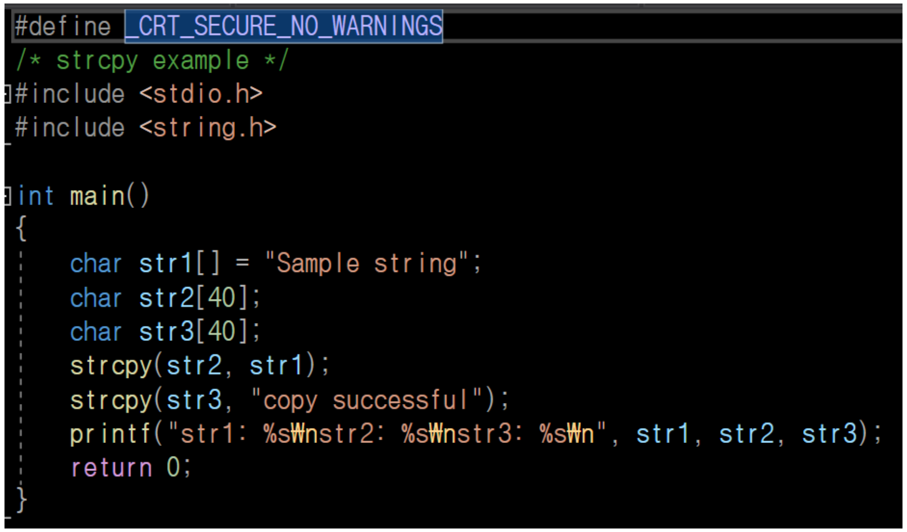
```
Looking for R2 astromech droids...
found R2D2
found R2A6
```

Assignment 2-1. 3

- **strcpy**
  - char *strcpy (char *destination, const char *source);
  - Copies the C string pointed by source into the array pointed by destination, including the terminating null character (and stopping at that point)
  - return value
    - *destination* is returned.

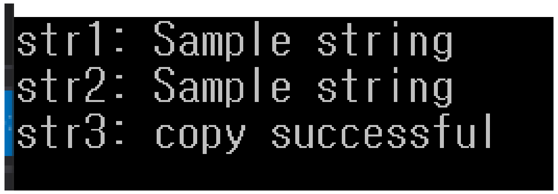
- **strncpy**
  - char * strncpy (char * destination, const char * source, size_t num);
  - Copies the first *num* characters of *source* to *destination*
  - If the end of the *source* C string (which is signaled by a null-character) is found before *num* characters have been copied, *destination* is padded with zeros until a total of *num* characters have been written to it
  - return value
    - *destination* is returned.

Assignment 2-1. 3

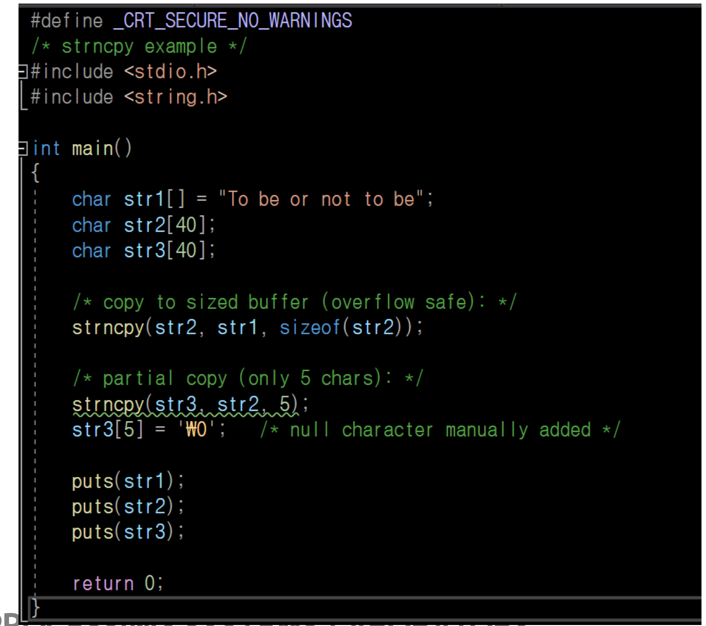


```c
#define _CRT_SECURE_NO_WARNINGS
/* strcpy example */
#include <stdio.h>
#include <string.h>

int main()
{
    char str1[] = "Sample string";
    char str2[40];
    char str3[40];
    strcpy(str2, str1);
    strcpy(str3, "copy successful");
    printf("str1: %s₩nstr2: %s₩nstr3: %s₩n", str1, str2, str3);
    return 0;
}
```

출력

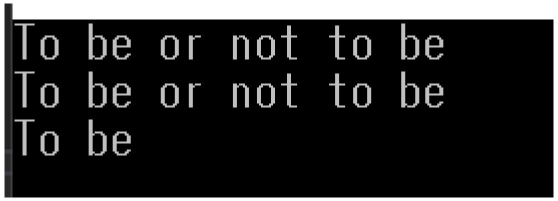


```
str1: Sample string
str2: Sample string
str3: copy successful
```



```c
#define _CRT_SECURE_NO_WARNINGS
/* strncpy example */
#include <stdio.h>
#include <string.h>

int main()
{
    char str1[] = "To be or not to be";
    char str2[40];
    char str3[40];

    /* copy to sized buffer (overflow safe): */
    strncpy(str2, str1, sizeof(str2));

    /* partial copy (only 5 chars): */
    strncpy(str3, str2, 5);
    str3[5] = '₩0';   /* null character manually added */

    puts(str1);
    puts(str2);
    puts(str3);

    return 0;
}
```

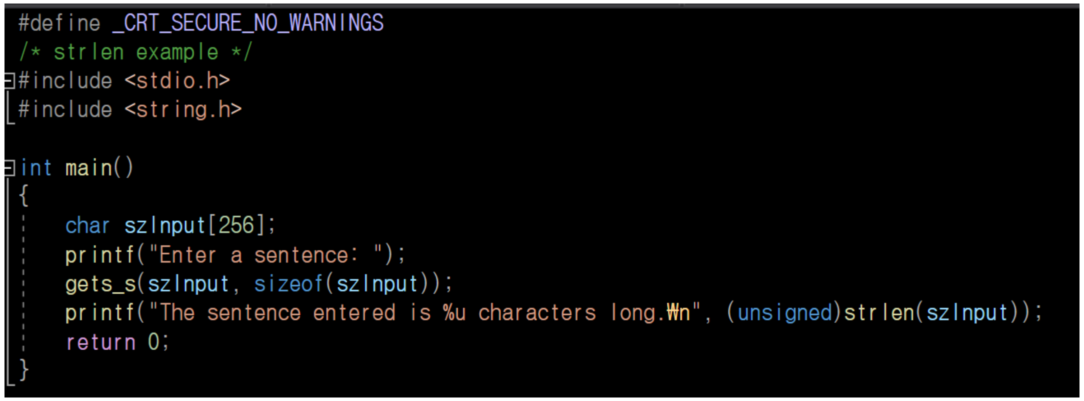
출력



```
To be or not to be
To be or not to be
To be
```

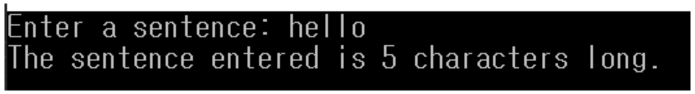
Assignment 2-1. 3

- **strlen**
  - size_t strlen (const char *str);
    - The length of a C string is determined by the terminating null-character
  - return value
    - Returns the length of the C string *str*.



```
#define _CRT_SECURE_NO_WARNINGS
/* strlen example */
#include <stdio.h>
#include <string.h>

int main()
{
    char szInput[256];
    printf("Enter a sentence: ");
    gets_s(szInput, sizeof(szInput));
    printf("The sentence entered is %u characters long.₩n", (unsigned)strlen(szInput));
    return 0;
}
```

출력



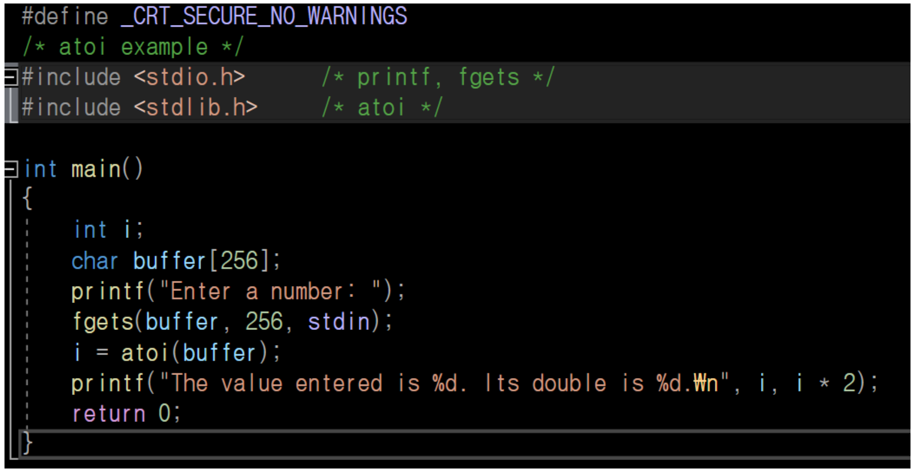
```
Enter a sentence: hello
The sentence entered is 5 characters long.
```

Assignment 2-1. 3

- **atoi**
  - int atoi (const char * str);
  - Parses the C-string *str* interpreting its content as an integral number, which is returned as a value of type int.
  - return value
    - On success, the function returns the converted integral number as an int value.

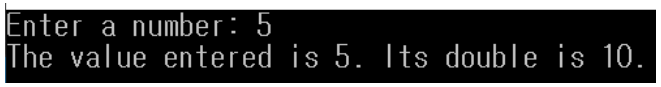
- **atof**
  - double atof (const char* str);
  - Parses the C string *str*, interpreting its content as a floating point number and returns its value as a double.
  - return value
    - On success, the function returns the converted floating point number as a double value.

Assignment 2-1. 3

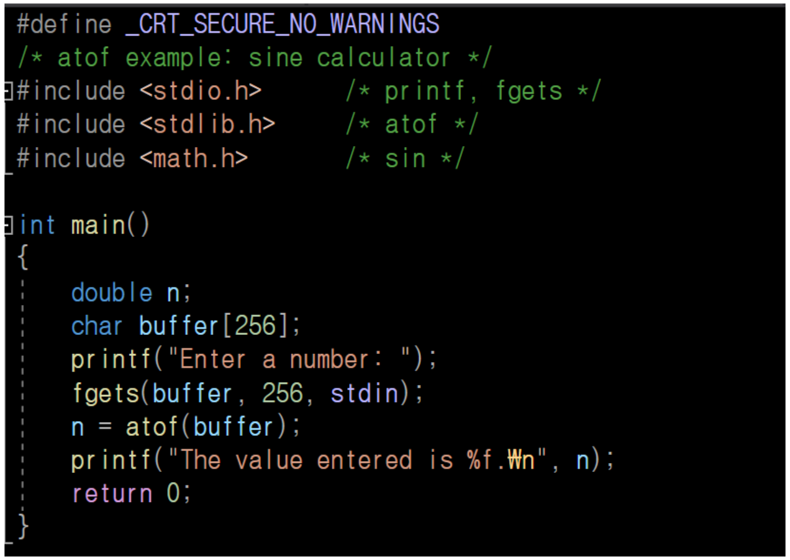


```
#define _CRT_SECURE_NO_WARNINGS
/* atoi example */
#include <stdio.h>      /* printf, fgets */
#include <stdlib.h>     /* atoi */

int main()
{
    int i;
    char buffer[256];
    printf("Enter a number: ");
    fgets(buffer, 256, stdin);
    i = atoi(buffer);
    printf("The value entered is %d. Its double is %d.₩n", i, i * 2);
    return 0;
}
```

출력

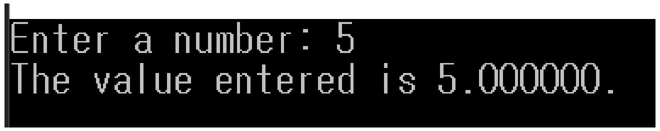


```
Enter a number: 5
The value entered is 5. Its double is 10.
```



```
#define _CRT_SECURE_NO_WARNINGS
/* atof example: sine calculator */
#include <stdio.h>      /* printf, fgets */
#include <stdlib.h>     /* atof */
#include <math.h>       /* sin */

int main()
{
    double n;
    char buffer[256];
    printf("Enter a number: ");
    fgets(buffer, 256, stdin);
    n = atof(buffer);
    printf("The value entered is %f.₩n", n);
    return 0;
}
```

출력



```
Enter a number: 5
The value entered is 5.000000.
```

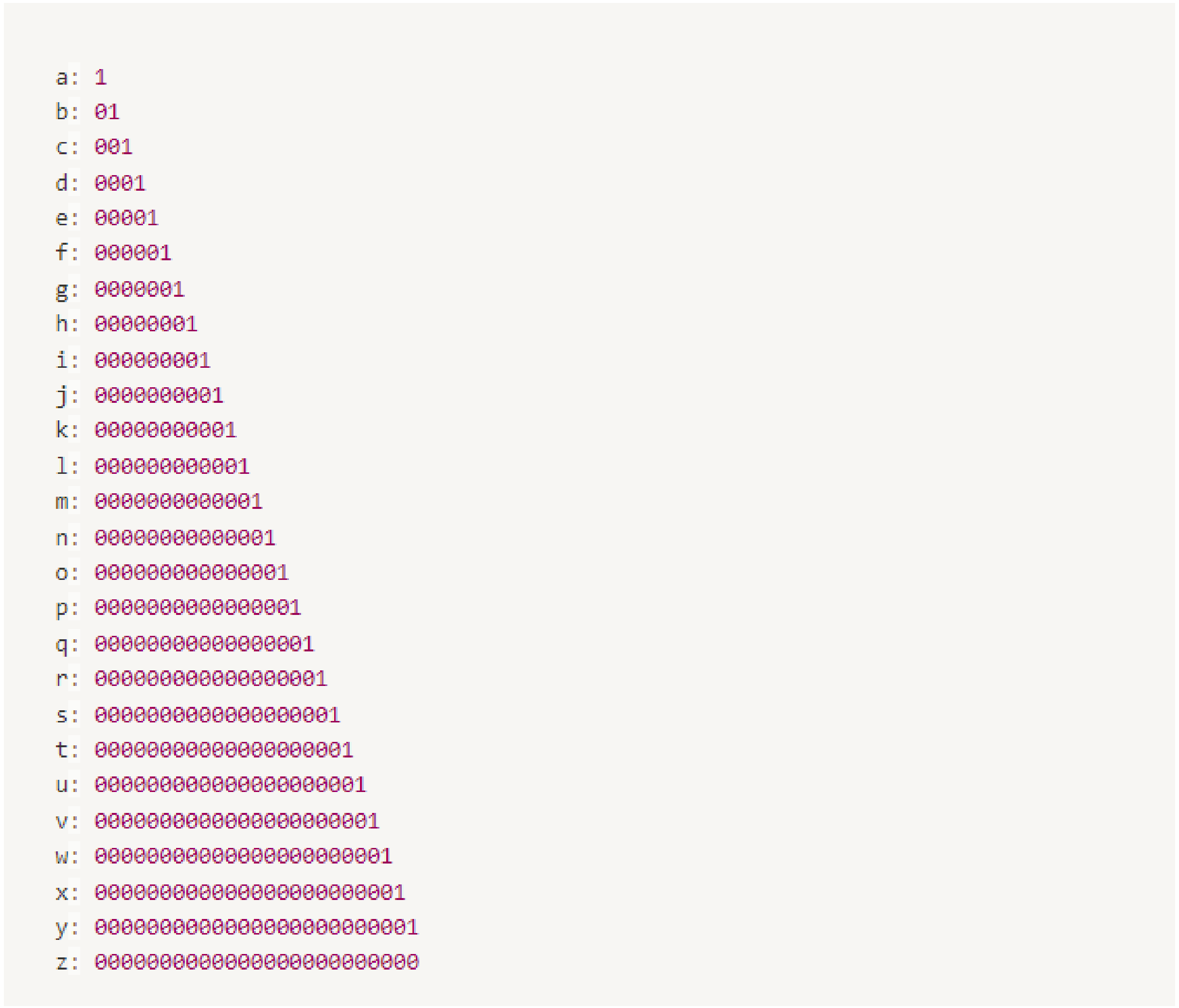
ASSIGNMENT 2-1. 4

Assignment 2-1. 4

- (Simple Text Parsing) Write a class, decode which is decoding binary stream to alphabets. Decoding procedure is as following.
  - Step 1. Read the binary file, (binary.txt)
  - Step 2. Decode binary file to get the characters.
  - Step 3. Write the alphabets which are decoded to the file

    (alphabet.txt)

Assignment 2-1. 4



```
a: 1
b: 01
c: 001
d: 0001
e: 00001
f: 000001
g: 0000001
h: 00000001
i: 000000001
j: 0000000001
k: 00000000001
l: 000000000001
m: 0000000000001
n: 00000000000001
o: 000000000000001
p: 0000000000000001
q: 00000000000000001
r: 000000000000000001
s: 0000000000000000001
t: 00000000000000000001
u: 000000000000000000001
v: 0000000000000000000001
w: 00000000000000000000001
x: 000000000000000000000001
y: 0000000000000000000000001
z: 0000000000000000000000000
```

Assignment 2-1. 4

- EX)

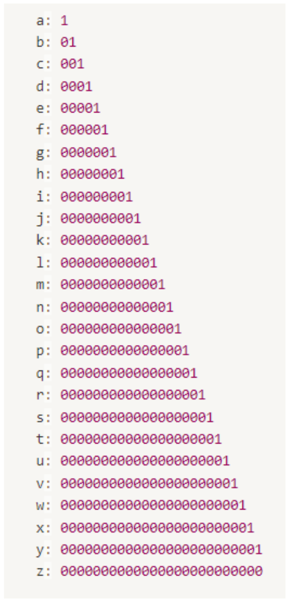
binary.txt

```
00000001000010000000000001000000000000100000000000000001
```

h e l l o

alphabet.txt

```
hello
```



```
a: 1
b: 01
c: 001
d: 0001
e: 00001
f: 000001
g: 0000001
h: 00000001
i: 000000001
j: 0000000001
k: 00000000001
l: 000000000001
m: 0000000000001
n: 00000000000001
o: 000000000000001
p: 0000000000000001
q: 00000000000000001
r: 000000000000000001
s: 0000000000000000001
t: 00000000000000000001
u: 000000000000000000001
v: 0000000000000000000001
w: 00000000000000000000001
x: 000000000000000000000001
y: 0000000000000000000000001
z: 0000000000000000000000000
```

과제 제출 방법

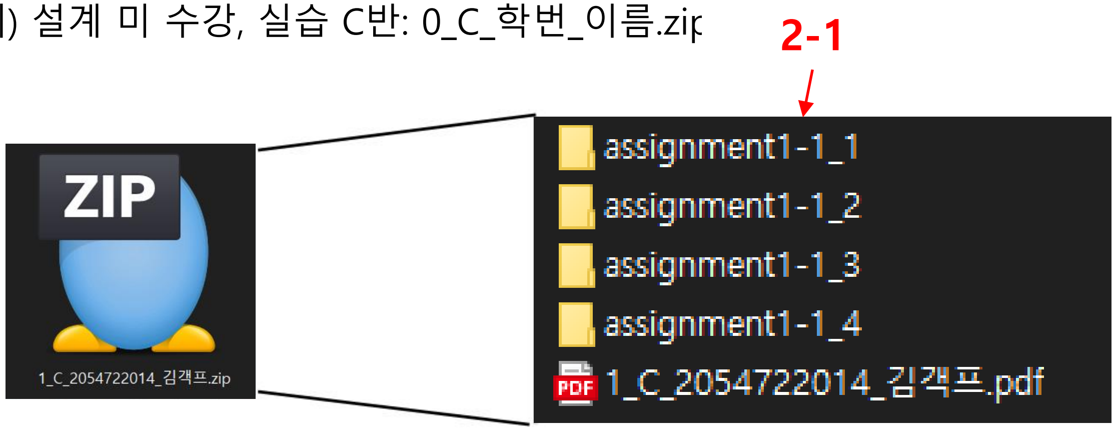
과제 제출 방법

- **FTP Upload (Klas 과제 제출 X)**
  - Address : ftp://223.194.8.1:1321
  - username : IPSL_OBJ
  - password : ipslobj_2023

- **Due date**
  - Soft copy: 마감일 **4/7(금)** 23:59:59까지 제출 (서버시간 기준)
  - Delay
    - 마감일 이후 +7일까지 제출 가능
    - 단, 1일 초과마다 과제 총점의 10%씩 감점

과제 제출 방법

- Soft copy
  - 과제(보고서, 소스 코드)를 압축한 파일 제출
    - 설계반_실습반_학번_이름.zip
      - 예) 설계1반 수강, 실습 A반: 1_A_학번_이름.zip
      - 예) 설계 수강, 실습 미수강: 2_0_학번_이름.zip
      - 예) 설계 미 수강, 실습 C반: 0_C_학번_이름.zip



1_C_2054722014_김갞프.zip

assignment1-1_1
assignment1-1_2
assignment1-1_3
assignment1-1_4
1_C_2054722014_김갞프.pdf

  - 과제 수정하여 업로드 시 버전 명시
    - 설계반_실습반_학번_이름_verX.zip

과제 제출 방법

- Soft copy
  - 과제 보고서
    - 영문 또는 한글로 작성
    - **반드시 PDF**로 제출 (PDF 외 파일 형식으로 제출시 0점 처리)
    - 보고서 양식
      - 문제 및 설명(문제 capture 금지) / 결과 화면 / 고찰
      - 보고서 양식은 아래 경로에서 참고
        - https://www.ipsl.kw.ac.kr/post/1%EC%B0%A8-%EA%B3%BC%EC%A0%9C
    - **소스코드 제외**
    - 분량 제한 없음
    - **표절 적발 시 0점 처리**
  - 소스 코드
    - Visual Studio 2022 community 사용 필수
      - https://docs.microsoft.com/ko-kr/visualstudio/install/install-visual-studio?view=vs-2022
    - STL (Standard Template Library) 사용 금지 (vector, map, algorithm 등)
    - Debug 폴더를 제외한 모든 파일 제출
      - .sln 파일 포함(.cpp 만 제출하지 말것)
    - **각 문제마다 프로젝트 파일 생성 필수**
    - **주석 반드시 달기**
    - **소스코드 표절 적발 시 0점 처리**

END OF PRESENTATION

Q&A