4190.103A-001: Programming Practice Lab. 11

Euidong Lee
Junseok Lee
Minhyo Jeong
(snucsl.ta@gmail.com)

Systems Software & Architecture Lab.

Seoul National University

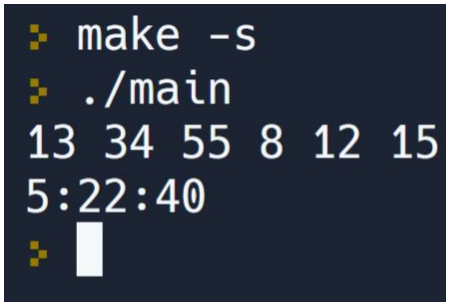
Spring 2022

과제 풀이

Lab 10 과제 1

Lab 10 과제 1

- 두 개의 시간을 입력 받아서 그 차이를 출력하는 프로그램을 작성하라.
- 입력 : 정수 6개를 순서대로 받는다
  - 이는 각각 다음에 해당한다: 시간1 분1 초1 시간2 분2 초2
  - 예시: 13 34 55 8 12 15 (시간1 13:34:55, 시간2 8:12:15)
- 출력 : 시간1과 시간2의 차이 (시분초는 콜론(:)으로 구분)
  - 예시: 5:22:40
  - 조건1: 사용자 입력을 main 함수 이외의 함수로 전달해 시간을 계산할 것
  - 조건2: 24시간제로 입력을 받는다.
  - 조건3: 처음 들어오는 시간이 나중에 들어오는 시간 보다 크다고 가정한다.
  - 조건4: 분, 초의 경우 10 이하일 경우 두 자리로 출력한다. (%02d 사용)



Lab 10 과제 1 - 해설

- main 함수
  - 시작, 끝, 차이를 저장하는 3가지 배열 선언
  - 배열에 입력을 받아 함수에 넘겨줌
  - 함수가 계산한 결과를 출력

```c
int main() {

    int start[3]; // int [hours][minutes][seconds]
    int stop[3];
    int diff[3];

    scanf("%d %d %d", &start[0], &start[1], &start[2]);
    scanf("%d %d %d", &stop[0], &stop[1], &stop[2]);

    differenceBetweenTimePeriod(start, stop, diff);
    printf("%d:%02d:%02d\n", diff[0], diff[1], diff[2]);
    return 0;
}
```

Lab 10 과제 1 - 해설

- differenceBetweenTimePeriod 함수
  - 시작 시간, 끝 시간, 차이를 저장할 배열을 받아옵니다
  - 낮은 자리수부터 순서대로 계산합니다
  - 시간의 단위가 60을 기준으로 변하는 것에 주의합니다

```c
#include <stdio.h>


void differenceBetweenTimePeriod(int *start, int *stop, int *diff) {
  while (stop[2] > start[2]) {
    --start[1];
    start[2] += 60;
  }

  diff[2] = start[2] - stop[2];

  while (stop[1] > start[1]) {
    --start[0];
    start[1] += 60;
  }
  diff[1] = start[1] - stop[1];

  diff[0] = start[0] - stop[0];
}
```

Lab 10 과제 1 - 해설

- 학생 답안

```
19 ▼ void differenceBetweenTimePeriod(int *start, int *stop, int *diff) {
20     diff[0] = start[0] - stop[0];
21     diff[1] = start[1] - stop[1];
22     diff[2] = start[2] - stop[2];
23 ▼   if(diff[2] < 0){
24       diff[2] += 60;
25       diff[1] -= 1;
26     }
27 ▼   if(diff[1] < 0){
28       diff[1] += 60;
29       diff[0] -= 1;
30     }
31   }
```

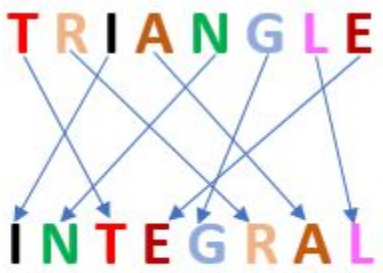
Lab 10 과제 1 - 해설

- 학생 답안

```
19 ▼ void differenceBetweenTimePeriod(int *start, int *stop, int *diff) {
20 ▼   for (int i = 0; i < 3; i++) {
21       diff[i] = start[i] - stop[i];
22
23 ▼     if ((i != 0) && (diff[i] < 0)) {
24         diff[i] += 60;
25         diff[i-1] -= 1;
26       }
27     }
28   }
```

Lab 10 과제 2

Lab 10 과제 2



- Anagram 이란?
  - 단어나 문장을 구성하고 있는 문자의 순서를 바꾸어 다른 단어나 문장을 만드는 놀이
- 입력: 개행으로 구분된 2개의 문자열
  - 각 문자열에는 공백 및 특수문자가 포함될 수 있다.
  - a~z, A~Z 범위 외 character는 입력으로 받되, Anagram 판별에 고려하지 않는다.
  - 각 문자열의 최대 길이는 null character 포함하여 64이다.
- 출력: Anagram이면 “true” 아니면 “false” 출력
  - “false ”와 “false”, “true ”와 “true”는 다른 문자열입니다!

Lab 10 과제 2 - 해설

- main 함수
  - 입력을 받을 2개의 배열 선언
  - 2개의 문자열 입력
  - 판단을 위해 isAnagram함수 호출

```c
#define MAX (64)
#define ALAPHABET_LENGTH (26)
#include <stdio.h>

int isAnagram(char str1[], char str2[]);

int main() {
  char arr1[MAX], arr2[MAX];

  scanf(" %[^\n]", arr1);
  scanf(" %[^\n]", arr2);
  isAnagram(arr1, arr2);
  return 0;
}
```

Lab 10 과제 2 - 해설

- getIndex 함수
  - 문자의 ascii 값을 알파벳 index로 변환

```
16 ▼ int getIndex(char c) {
17 ▼   if ((c >= 'a' && c <= 'z')) {
18       return c - 'a';
19     }
20     return c - 'A';
21   }
```

Lab 10 과제 2 - 해설

- printAnswer 함수
  - 각 문자열에서 등장한 알파벳들의 수가 들어있는 두 배열을 받아옴
  - 두 배열을 비교해서 “true”, “false”를 출력

```
23 ▼ void printAnswer(int a[], int b[]) {
24 ▼   for (int i = 0; i < ALAPHABET_LENGTH; i++) {
25 ▼     if (a[i] != b[i]){
26         printf("false\n");
27         return;
28       }
29     }
30     printf("true\n");
31   }
```

Lab 10 과제 2 - 해설

- isAnagram 함수
  - 각 알파벳이 등장하는 횟수를 저장할 배열 2개
  - 알파벳의 index와 대응되는 칸이 해당 알파벳이 등장한 횟수를 저장

```
33 int isAnagram(char str1[], char str2[]) {
34   int alphabet1[ALAPHABET_LENGTH] = {
35       0,
36   };
37   int alphabet2[ALAPHABET_LENGTH] = {
38       0,
39   };
40
41   for (int i = 0; i < strlen(str1) + 1; i++) {
42     if ((str1[i] >= 'a' && str1[i] <= 'z') ||
43         (str1[i] >= 'A' && str1[i] <= 'Z')) {
44       alphabet1[getIndex(str1[i])]++;
45     }
46
47   }
48
49   for (int i = 0; i < strlen(str2) + 1; i++) {
50     if ((str2[i] >= 'a' && str2[i] <= 'z') ||
51         (str2[i] >= 'A' && str2[i] <= 'Z')) {
52       alphabet2[getIndex(str2[i])]++;
53     }
54   }
55
56   printAnswer(alphabet1, alphabet2);
57
58   return 0;
59 }
```

Lab 10 과제 2 - 해설

- isAnagram 함수
  - 첫번째 문자열을 훑으며, 각 알파벳마다 등장 횟수를 count

```
33 ▼ int isAnagram(char str1[], char str2[]) {
34 ▼   int alphabet1[ALAPHABET_LENGTH] = {
35         0,
36     };
37 ▼   int alphabet2[ALAPHABET_LENGTH] = {
38         0,
39     };
40
41 ▼   for (int i = 0; i < strlen(str1) + 1; i++) {
42       if ((str1[i] >= 'a' && str1[i] <= 'z') ||
43 ▼         (str1[i] >= 'A' && str1[i] <= 'Z')) {
44         alphabet1[getIndex(str1[i])]++;
45       }
46
47     }
48
49 ▼   for (int i = 0; i < strlen(str2) + 1; i++) {
50       if ((str2[i] >= 'a' && str2[i] <= 'z') ||
51 ▼         (str2[i] >= 'A' && str2[i] <= 'Z')) {
52         alphabet2[getIndex(str2[i])]++;
53       }
54     }
55
56     printAnswer(alphabet1, alphabet2);
57
58     return 0;
59 }
```

Lab 10 과제 2 - 해설

- isAnagram 함수
  - 두번째 문자열을 훑으며, 각 알파벳마다 등장 횟수를 count

```
33 int isAnagram(char str1[], char str2[]) {
34   int alphabet1[ALAPHABET_LENGTH] = {
35       0,
36   };
37   int alphabet2[ALAPHABET_LENGTH] = {
38       0,
39   };
40
41   for (int i = 0; i < strlen(str1) + 1; i++) {
42     if ((str1[i] >= 'a' && str1[i] <= 'z') ||
43         (str1[i] >= 'A' && str1[i] <= 'Z')) {
44       alphabet1[getIndex(str1[i])]++;
45     }
46
47   }
48
49   for (int i = 0; i < strlen(str2) + 1; i++) {
50     if ((str2[i] >= 'a' && str2[i] <= 'z') ||
51         (str2[i] >= 'A' && str2[i] <= 'Z')) {
52       alphabet2[getIndex(str2[i])]++;
53     }
54   }
55
56   printAnswer(alphabet1, alphabet2);
57
58   return 0;
59 }
```

Lab 10 과제 2 - 해설

- isAnagram 함수
  - Count가 끝난 두 배열을 printAnswer 함수를 호출하며 전달함

```
33 ▼ int isAnagram(char str1[], char str2[]) {
34 ▼   int alphabet1[ALAPHABET_LENGTH] = {
35         0,
36     };
37 ▼   int alphabet2[ALAPHABET_LENGTH] = {
38         0,
39     };
40
41 ▼   for (int i = 0; i < strlen(str1) + 1; i++) {
42       if ((str1[i] >= 'a' && str1[i] <= 'z') ||
43 ▼         (str1[i] >= 'A' && str1[i] <= 'Z')) {
44         alphabet1[getIndex(str1[i])]++;
45       }
46
47     }
48
49 ▼   for (int i = 0; i < strlen(str2) + 1; i++) {
50       if ((str2[i] >= 'a' && str2[i] <= 'z') ||
51 ▼         (str2[i] >= 'A' && str2[i] <= 'Z')) {
52         alphabet2[getIndex(str2[i])]++;
53       }
54     }
55
56     printAnswer(alphabet1, alphabet2);
57
58     return 0;
59 }
```

Lab 10 과제 2 - 해설

- 학생 답안

```c
int isAnagram(char str1[], char str2[]) {
    int cnt[30] = {0,};

    for (int i = 0; str1[i]; i++) {
        char c = str1[i];
        if ('A' <= c && c <= 'Z') cnt[c - 'A']++;
        else if ('a' <= c && c <= 'z') cnt[c - 'a']++;
    }

    for (int i = 0; str2[i]; i++) {
        char c = str2[i];
        if ('A' <= c && c <= 'Z') cnt[c - 'A']--;
        else if ('a' <= c && c <= 'z') cnt[c - 'a']--;
    }

    for (int i = 0; i < 26; i++) {
        if (cnt[i] != 0) {
            printf("false\n");
            return RETURN_ERROR;
        }
    }

    printf("true\n");
    return 0;
}
```

실습 1

실습 1

- 포인터에 익숙해지기 위해서 다음 문제를 같이 풀어봅시다!!
- 포인터를 사용하는 코드가 주어집니다
- 해당 프로그램을 실행했을 때, 콘솔에 출력되는 결과를 맞춰보세요!
- 코드를 쳐보지 말고 머리속으로 계산해보세요

실습 1

문제 1

```c
#include <stdio.h>

int main(){
    int a, b;
    int *ptr;

    a = 3;
    b = 4;
    ptr = &a;
    b = *ptr;

    printf("%d\n", b);

    return 0;
}
```

문제 2

```c
#include <stdio.h>

int main(){
    int a, b;
    int *a_ptr;
    int *b_ptr;

    a = 3;
    b = 4;
    a_ptr = &a;
    b_ptr = &b;
    *b_ptr = 3;

    printf("%d\n", b);

    return 0;
}
```

문제 3

```c
#include <stdio.h>

int main(){
    int a, b;
    int *a_ptr;
    int *b_ptr;

    a = 3;
    b = 4;
    a_ptr = &a;
    b_ptr = &b;
    *a_ptr = *b_ptr;

    printf("%d\n", a);

    return 0;
}
```

실습 1

문제 4

```c
#include <stdio.h>

int main(){
    int a, b;
    int *a_ptr;
    int *b_ptr;

    a = 3;
    b = 4;
    a_ptr = &a;
    b_ptr = &b;
    a_ptr = b_ptr;
    *a_ptr = 5;

    printf("%d\n", b);

    return 0;
}
```

문제 5

```c
#include <stdio.h>

int main(){
    int a, b;
    int *a_ptr;
    int *b_ptr;

    a = 3;
    b = 4;
    a_ptr = &a;
    b_ptr = &b;
    *a_ptr = 6;
    b_ptr = a_ptr;

    printf("%d\n", b);

    return 0;
}
```

실습 1

문제 6

```c
#include <stdio.h>

void add_one(int a){
    a = a + 1;
}

int main(){
    int a, b;

    a = 3;
    b = 4;

    add_one(b);

    printf("%d\n", b);

    return 0;
}
```

문제 7

```c
#include <stdio.h>

void add_one(int *a){
    a = a + 1;
}

int main(){
    int a, b;

    a = 3;
    b = 4;

    add_one(&b);

    printf("%d\n", b);

    return 0;
}
```

문제 8

```c
#include <stdio.h>

void add_one(int *a){
    *a = *a + 1;
}

int main(){
    int a, b;

    a = 3;
    b = 4;

    add_one(&b);

    printf("%d\n", b);

    return 0;
}
```

실습 1

문제 9

```c
#include <stdio.h>

void add_one(int *a){
    *a = *a + 1;
}

int main(){
    int a, b;
    int *ptr;

    a = 3;
    b = 4;

    ptr = &b;
    add_one(ptr);

    printf("%d\n", b);

    return 0;
}
```

문제 10

```c
#include <stdio.h>

void func(int a, int b){
    a = a + b;
}

int main(){
    int a, b;

    a = 3;
    b = 4;

    func(a, b);

    printf("%d\n", a);

    return 0;
}
```

문제 11

```c
#include <stdio.h>

void func(int *a, int *b){
    *a = *a + *b;
}

int main(){
    int a, b;

    a = 3;
    b = 4;

    func(&b, &a);

    printf("%d\n", a);

    return 0;
}
```

Debugger

Debugger

- Debugger에 대해서 배워봅시다
  - GDB(GNU Debugger)
  - Replit Debugger

Debugger

- 프로그램을 작성하다 보면 무수히 많은 bug를 접하게 됩니다
  - 이게 왜 안되지?
  - 이건 왜 되지?
- 이런 bug들을 수정하는 것을 debugging이라고 합니다
  - Debugging을 도와주는 tool이 바로 debugger 입니다
- 동작 예시와 함께 배워봅시다

Debugger

- 디버깅 하고자 하는 프로그램

```c
#include <stdio.h>

void swap(int *num1, int *num2){
  int temp = *num1;
  *num1 = *num2;
  *num2 = temp;
}

int main(void) {
  int arr[10] = {2, 6, 3, 8, 4, 7, 5, 1, 0, 9};

  for(int i = 0; i < 9; i++){
    for(int j = i+1; j < 10; j++){
      if(arr[i] > arr[j])
        swap(arr+i, arr+j);
    }
  }

  return 0;
}
```

GDB

- GDB 실행
  - gcc로 작성한 프로그램의 컴파일 진행
    - -g : Debugging에 필요한 정보를 생성
    - -O0(대문자O + 숫자0) : 컴파일러 최적화 옵션(최적화 기법 사용 안함)

```
> gcc -g -O0 main.c
In file included from /nix/store/iwd8ic5hhwdxn5dga0im55g5hjl270cd-glibc-2.33-47-dev/include/bits/libc-header-start.h:33,
                 from /nix/store/iwd8ic5hhwdxn5dga0im55g5hjl270cd-glibc-2.33-47-dev/include/stdio.h:27,
                 from main.c:1:
/nix/store/iwd8ic5hhwdxn5dga0im55g5hjl270cd-glibc-2.33-47-dev/include/features.h:397:4: warning: #warning _FORTIFY_SOURCE requires compiling with optimization (-O) [-Wcpp]
  397 | #  warning _FORTIFY_SOURCE requires compiling with optimization (-O)
      |    ^~~~~~~
>
```

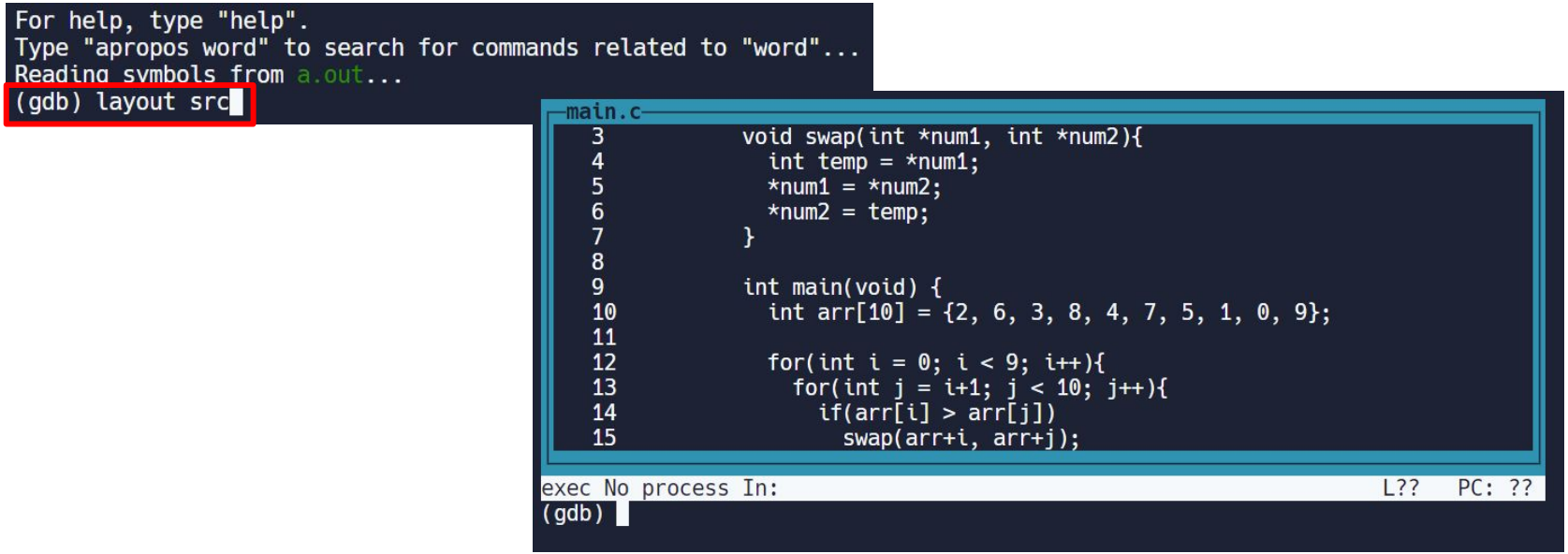
GDB

- GDB 실행
  - gdb a.out
    - a.out : 이전의 컴파일 과정으로 생성된 실행파일

```
$ gdb a.out
GNU gdb (GDB) 10.2
Copyright (C) 2021 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-unknown-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from a.out...
(gdb)
```

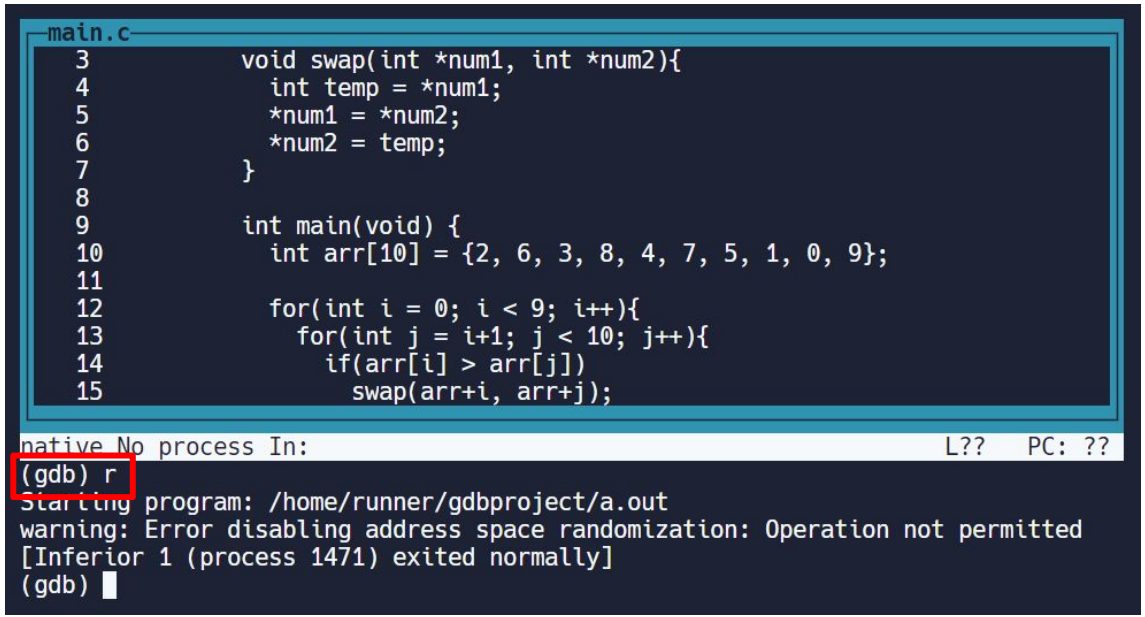
GDB

- GDB 사용 - code 보기
  - layout src



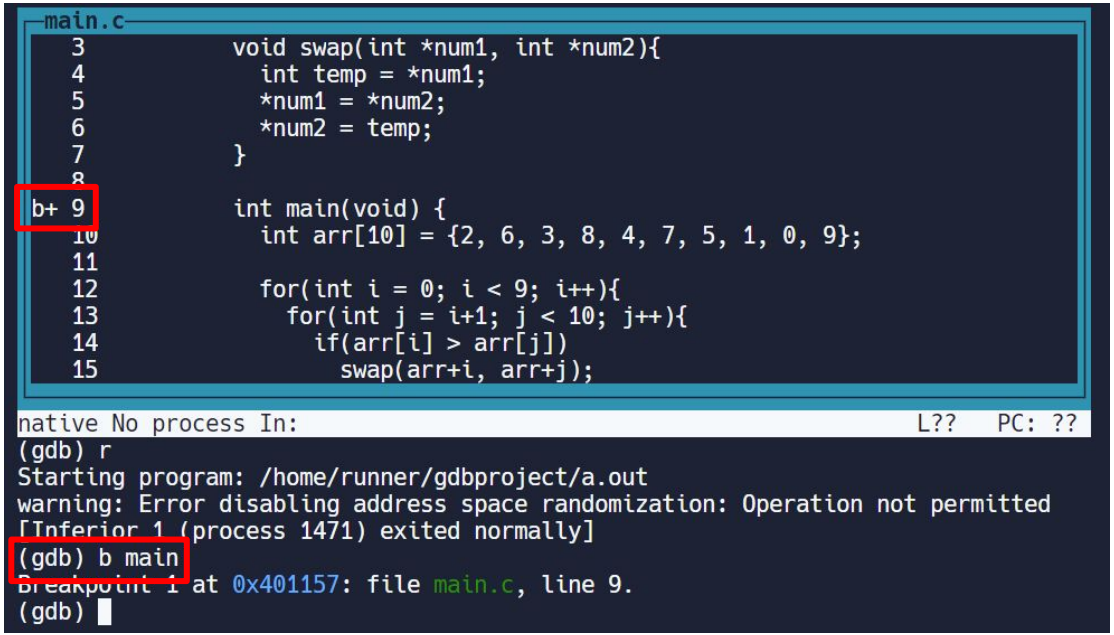
GDB

- GDB 사용 - 작성한 프로그램 실행
  - r(run)



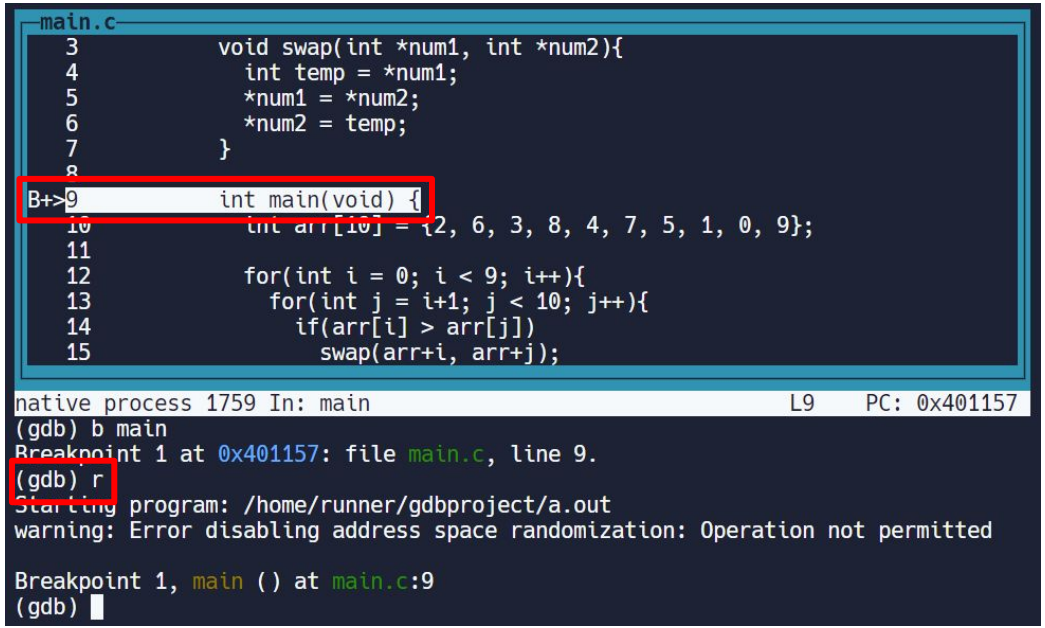
GDB

- GDB 사용 - 중단점(break point) 생성
  - b [line No. 또는 function name]



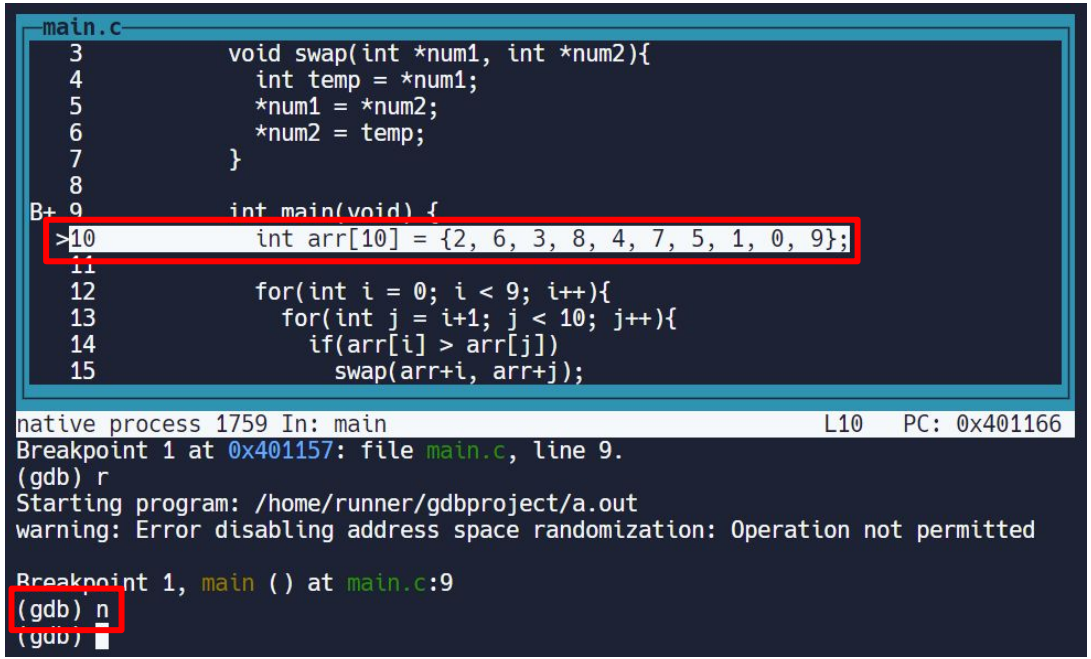
GDB

- 다시 실행
  - 중단점에서 프로그램이 멈춘 것을 확인 가능



GDB

- GDB 사용 - 다음 줄 실행
  - n(next)



GDB

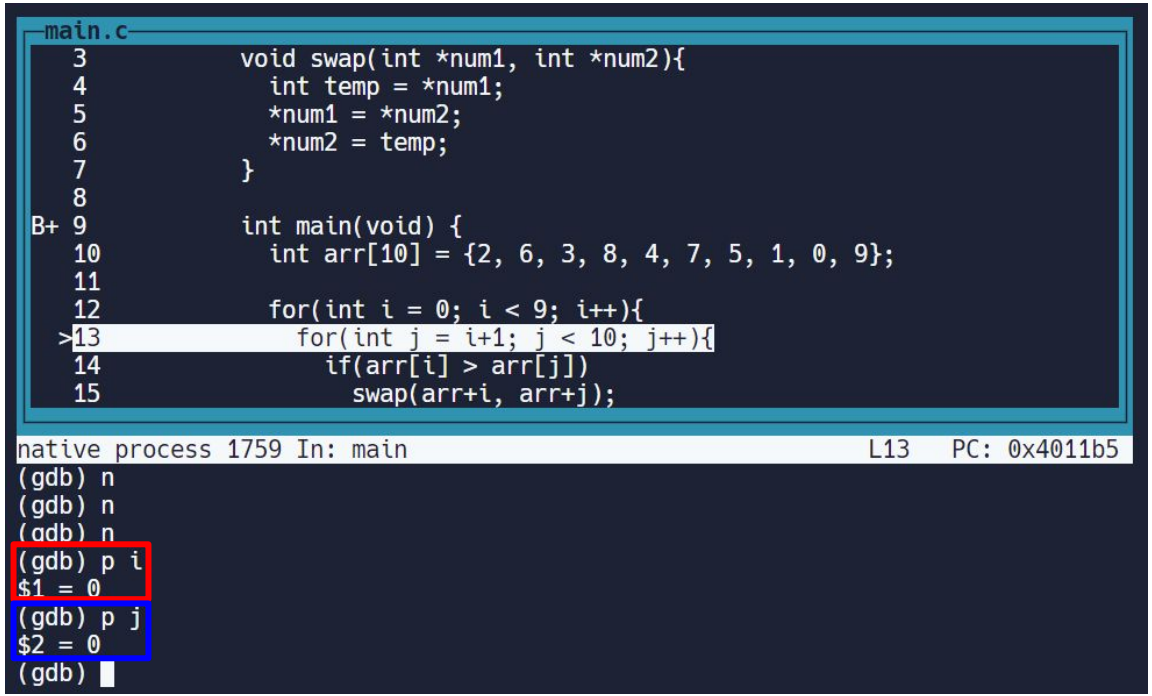
```
main.c
   3            void swap(int *num1, int *num2){
   4              int temp = *num1;
   5              *num1 = *num2;
   6              *num2 = temp;
   7            }
   8
B+ 9            int main(void) {
   10             int arr[10] = {2, 6, 3, 8, 4, 7, 5, 1, 0, 9};
   11
  >12               for(int i = 0; i < 9; i++){
   13                 for(int j = i+1; j < 10; j++){
   14                   if(arr[i] > arr[j])
   15                     swap(arr+i, arr+j);
native process 1759 In: main                                  L12   PC: 0x4011ac
(gdb) r
Starting program: /home/runner/gdbproject/a.out
warning: Error disabling address space randomization: Operation not permitted

Breakpoint 1, main () at main.c:9
(gdb) n
(gdb) n
(gdb)
```

GDB

```
main.c
   3            void swap(int *num1, int *num2){
   4              int temp = *num1;
   5              *num1 = *num2;
   6              *num2 = temp;
   7            }
   8
B+ 9            int main(void) {
  10              int arr[10] = {2, 6, 3, 8, 4, 7, 5, 1, 0, 9};
  11
  12              for(int i = 0; i < 9; i++){
 >13                for(int j = i+1; j < 10; j++){
  14                  if(arr[i] > arr[j])
  15                    swap(arr+i, arr+j);
native process 1759 In: main                                  L13   PC: 0x4011b5
Starting program: /home/runner/gdbproject/a.out
warning: Error disabling address space randomization: Operation not permitted

Breakpoint 1, main () at main.c:9
(gdb) n
(gdb) n
(gdb) n
(gdb)
```

GDB

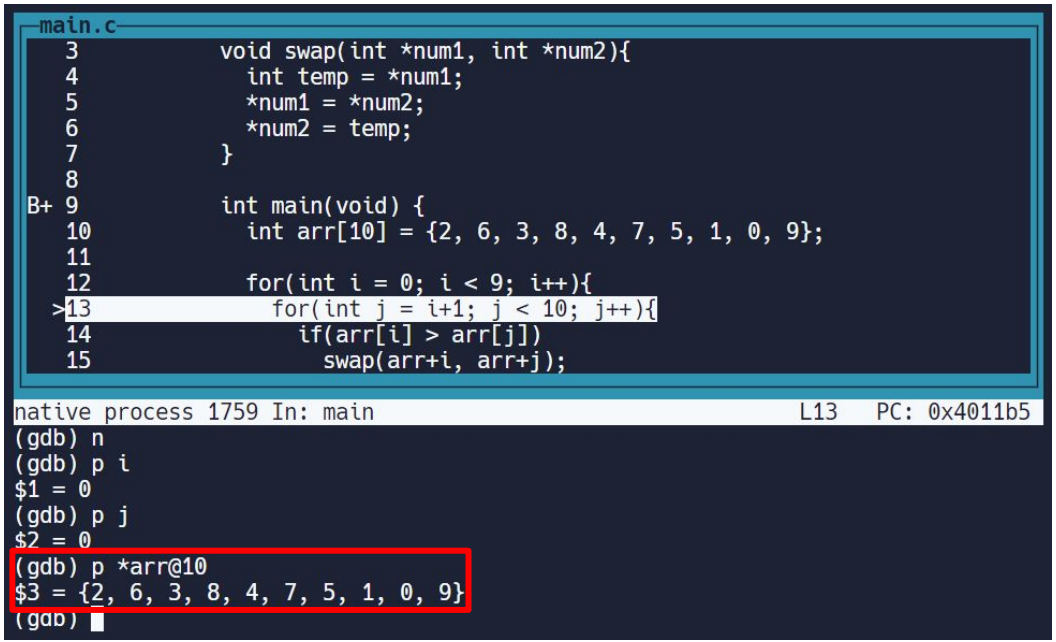
- GDB 사용 - 변수의 값 확인
  - p(print) [변수 명]



할당 전

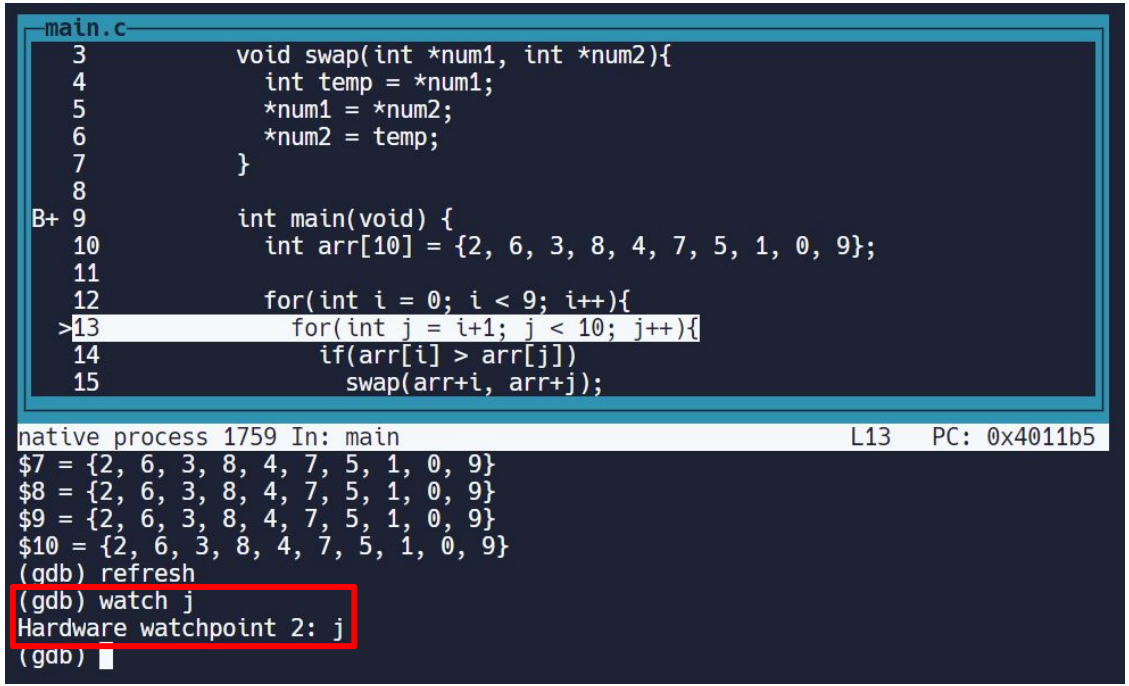
GDB

- GDB 사용 - 배열의 값 확인
  - p(print) *[배열 명]@[배열 길이]



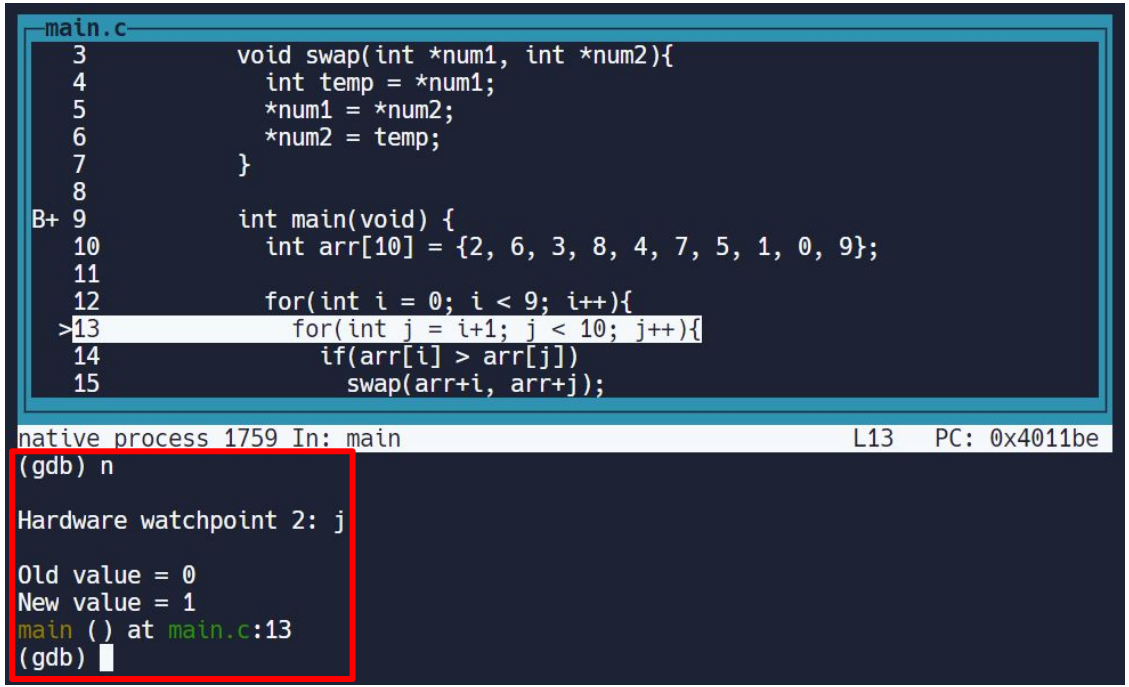
GDB

- GDB 사용 - 변수 값 추적(값이 변하면 자동으로 출력)
  - watch [변수 명]



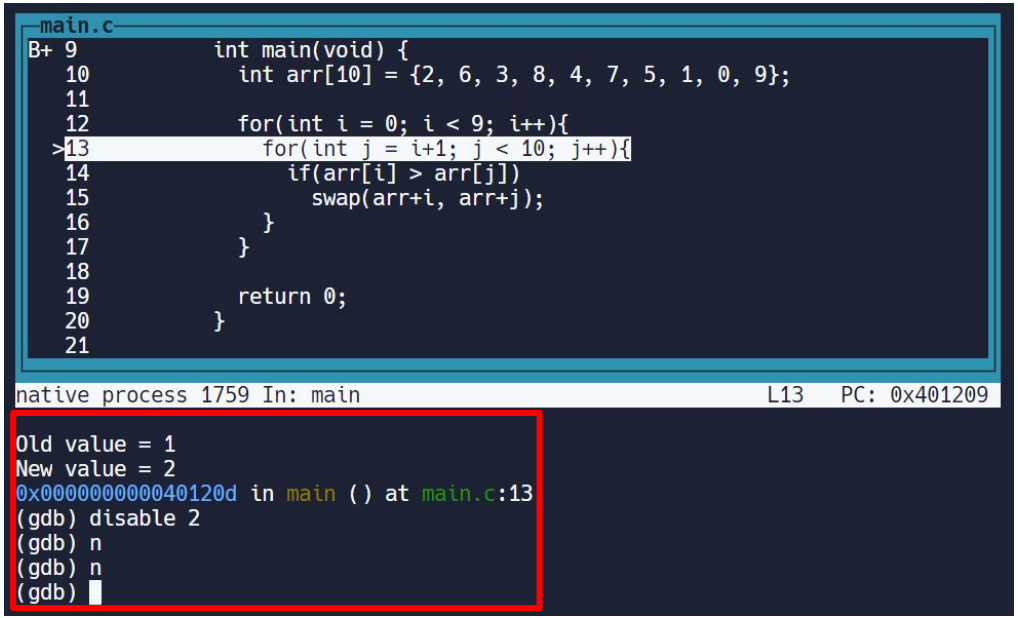
GDB

- GDB 사용 - 변수 값 추적(값이 변하면 자동으로 출력)
  - watch [변수 명]



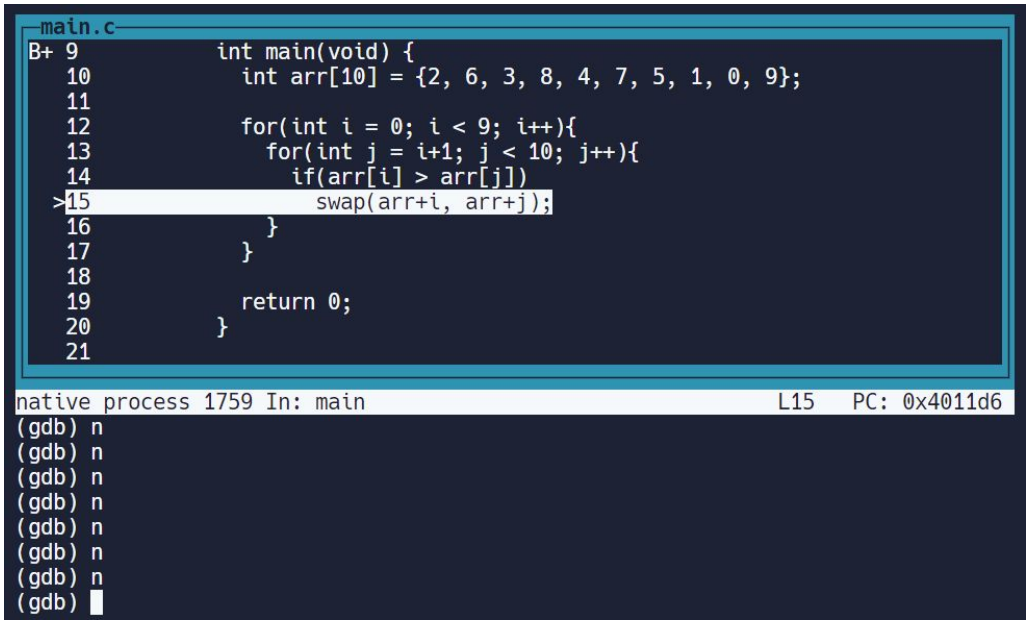
GDB

- GDB 사용 - watch 해제
  - disable [watchpoint 번호]



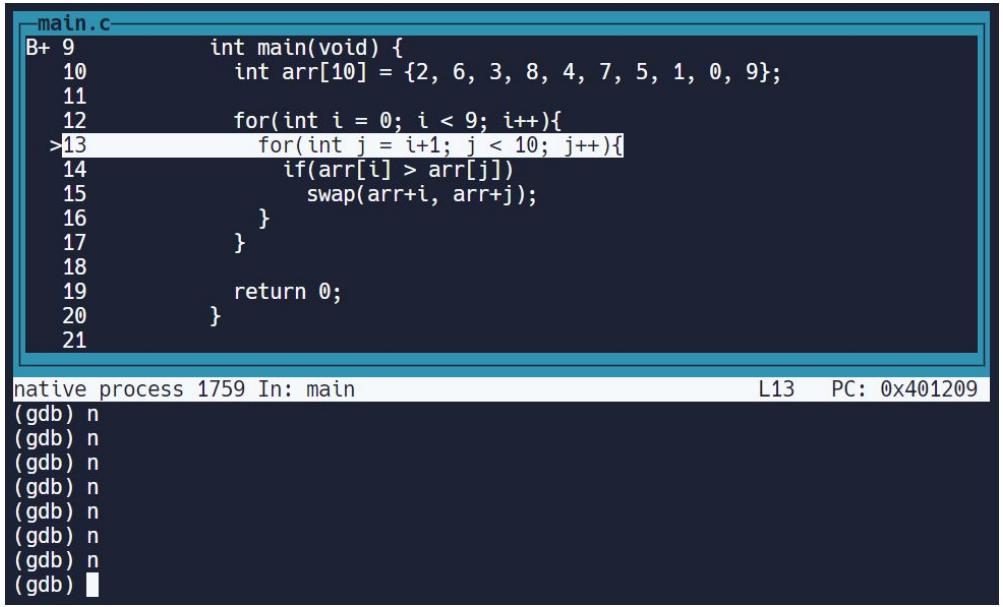
GDB

- GDB 사용 - 내부 코드로 진입
  - swap 함수를 수행할 차례



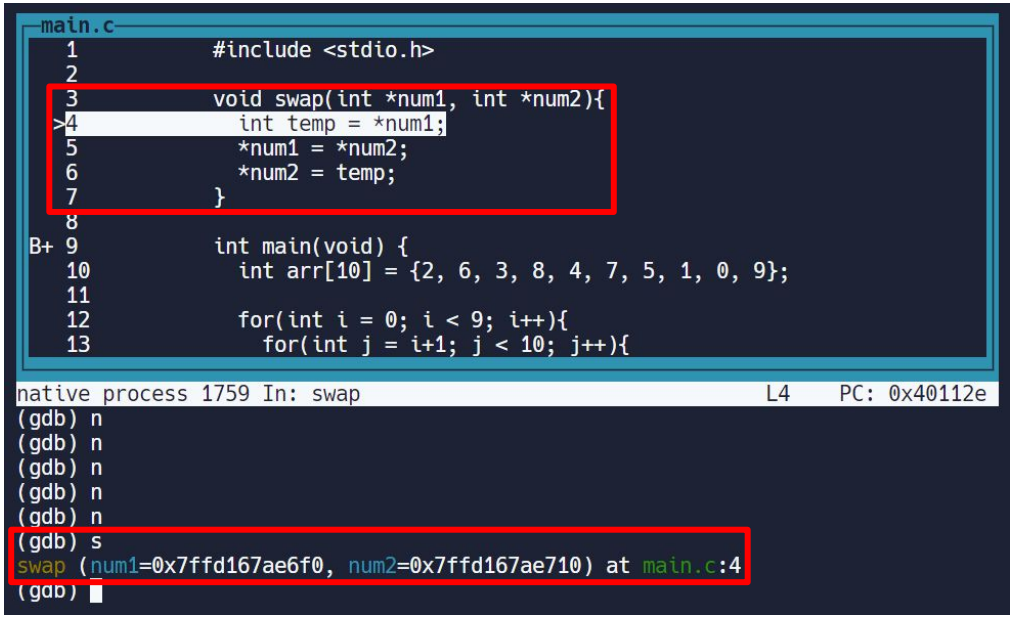
GDB

- GDB 사용 - 내부 코드로 진입
  - n을 입력하면, swap 함수 내부로 들어가지 않고 진행



GDB

- GDB 사용 - 내부 코드로 진입
  - s(step) 사용하면 함수 내부 코드로 진입 가능



GDB

- GDB 사용 - break point 정보 확인
  - info break

```
(gdb) info break
Num     Type           Disp Enb Address            What
1       breakpoint     keep y   0x0000000000401157 in main at main.c:9
        breakpoint already hit 1 time
2       hw watchpoint  keep n                      j
        breakpoint already hit 2 times
(gdb)
```

GDB

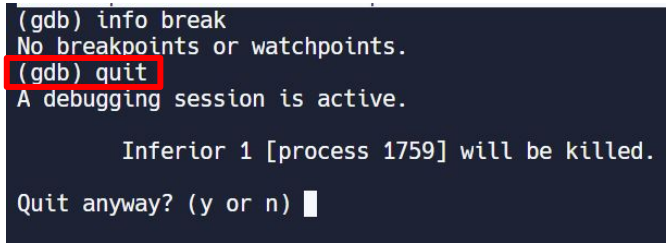
- GDB 사용 - break point 제거
  - d(delete) [번호]

```
Num     Type           Disp Enb Address            What
1       breakpoint     keep y   0x0000000000401157 in main at main.c:9
        breakpoint already hit 1 time
2       hw watchpoint  keep n                      j
        breakpoint already hit 2 times
(gdb) d 1
(gdb) delete 2
(gdb)
```

```
        breakpoint already hit 1 time
2       hw watchpoint  keep n                      j
        breakpoint already hit 2 times
(gdb) d 1
(gdb) delete 2
(gdb) info break
No breakpoints or watchpoints.
(gdb)
```

GDB

- GDB 사용 - 종료
  - quit



Replit Debugger

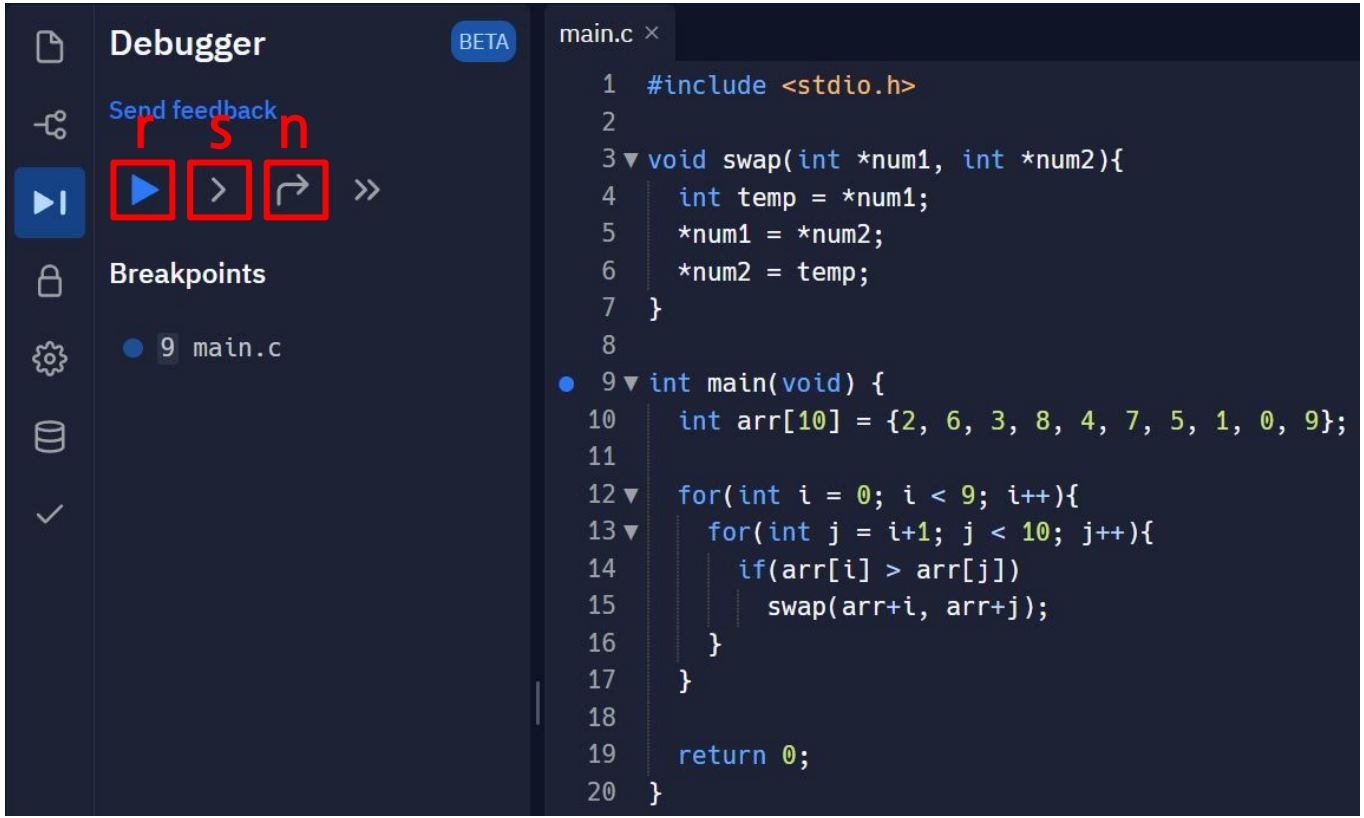
- 각 IDE들에는 Debugging 기능이 포함되어 있습니다
  - 사용법과 동작 원리는 비슷합니다

Replit Debugger

- 중단점은 원하는 line 번호 왼쪽을 클릭하시면 됩니다

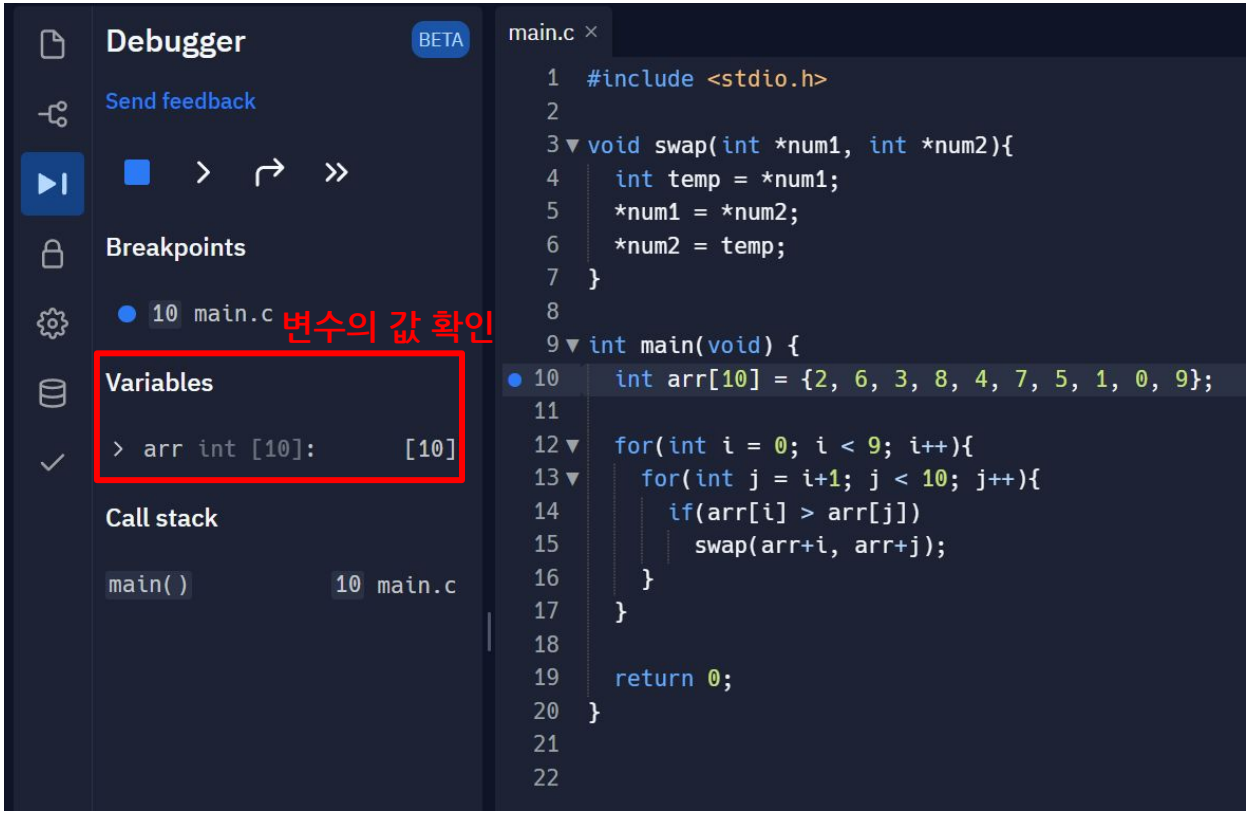
Replit Debugger

- 기능



Replit Debugger

- 실행 화면



실습 2

실습 2 - 설명

- 지금까지 배운 것을 총동원하여 프로젝트를 진행해봅시다!
  - Struct, bitwise operation, enum을 사용해봅시다
- 저희가 만들고자 하는 것은 학생 관리 프로그램입니다
- 프로그램은 다음의 기능을 갖고 있습니다
  - 학생들의 정보 저장
  - 학생들의 과제 제출 여부 저장
  - 학생들의 학점 계산 및 저장

실습 2 - 설명

- 먼저 학생을 등록하는 기능을 만들어봅시다!
  - 실습을 진행하면서 기능을 추가할 예정입니다

```
> make -s
> ./main
Choose option
(0: Exit, 1: Student Registration, 2: Show Students)
- 1
Name: jslee
ID: 2021-11111
Registration Complete
Choose option
(0: Exit, 1: Student Registration, 2: Show Students)
- 1
Name: euidong
ID: 2021-22222
Registration Complete
Choose option
(0: Exit, 1: Student Registration, 2: Show Students)
- 1
Name: minhyo
ID: 2021-33333
Registration Complete
Choose option
(0: Exit, 1: Student Registration, 2: Show Students)
- 2
---------
No.1
Name: jslee
ID: 2021-11111
No.2
Name: euidong
ID: 2021-22222
No.3
Name: minhyo
ID: 2021-33333
---------
Choose option
(0: Exit, 1: Student Registration, 2: Show Students)
-
```

실습 2 - 설명

- 학생의 정보를 저장하기 위해서 struct를 이용합니다
  - 저장할 정보는 이름과, 학번입니다
  - 둘 다 string의 형태로 저장합니다(이름은 10칸, 학번은 11칸)
- 만든 struct를 사용해봅시다!
  - main 함수 안에서 지역 변수로 struct 배열을 선언해보세요
  - 저장할 수 있는 학생은 최대 10명입니다
  - 등록된 총 학생 수를 나타내는 변수도 main 함수의 지역 변수로 추가합니다

실습 2 - 설명

- 코드 예시

```c
#include <stdio.h>

#define MAX_STD 10

struct student{
    char name[10];
    char id[11];
};

int main(){
    struct student stds[MAX_STD];
    int std_cnt = 0;

    return 0;
}
```

실습 2 - 설명

- 새로운 학생을 등록하는 함수(new_std)를 만들어봅시다
  - main 함수에서 만들어진 배열에 추가해야겠죠?
  - 등록이 완료된다면, 마찬가지로 main 함수에서 선언된 std_cnt 변수를 증가시켜야합니다
  - 포인터를 쓰고 싶지 않나요?
- new_std 함수는 다음 조건을 만족합니다
  - 배열이 가득 찼다면 0을 return 해서 등록을 실패했음을 알려줍니다
  - 정상적으로 저장이 되었다면 1을 return 합니다
  - scanf를 이용해서 새로 등록할 학생의 필요한 정보를 입력받습니다

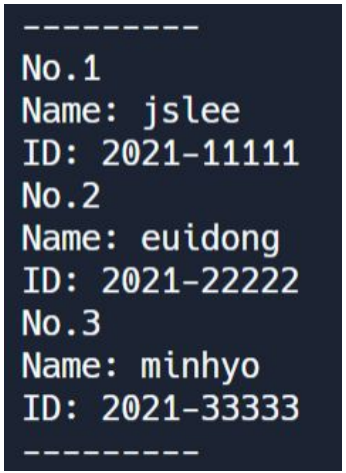
실습 2 - 설명

- 코드 예시

```c
int new_std(struct student stds[], int* std_cnt){
  if(*std_cnt == MAX_STD)
    return 0;

  printf("Name: ");
  scanf("%s", stds[*std_cnt].name);

  printf("ID: ");
  scanf("%s", stds[*std_cnt].id);

  (*std_cnt)++;
  return 1;
}
```

실습 2 - 설명

- 학생 정보가 잘 저장되었는지 확인하고 싶습니다!
- 저장된 학생 정보를 출력하는 함수(show_stds)를 만들어봅시다!
  - 배열에 저장된 순서대로 학생의 정보를 나열합니다



실습 2 - 설명

- 코드 예시

```c
void show_stds(struct student stds[], int std_cnt){
    printf("---------\n");
    for(int i = 0; i < std_cnt; i++){
        printf("No.%d\n", i+1);
        printf("Name: %s\n", stds[i].name);
        printf("ID: %s\n", stds[i].id);
    }
    printf("---------\n");
}
```

실습 2

- 이제 만든 함수들을 조합해서 학생 정보를 저장하는 프로그램을 완성해 봅시다!
  - main 함수에 필요한 logic을 채워넣어 봅시다

실습 2

- 먼저, 사용자에게 원하는 option을 입력 받는 기능을 구현합니다
  - 프로그램은 사용자가 끝내는 옵션을 선택할 때까지 반복합니다.
  - 따라서, 반복문 안에서 입력을 받을 수 있도록 작성해보세요!
- 옵션의 항목은 다음과 같습니다
  - 0: Exit, 1: Student Registration, 2: Show Students

실습 2

- 코드 예시

```
34 ▼ int main(){
35     struct student stds[MAX_STD];
36     int std_cnt = 0;
37     int option = -1;
38
39 ▼   while(option){
40       printf("Choose option\n");
41       printf("(0: Exit, 1: Student Registration, 2: Show Students)\n- ");
42       scanf("%d", &option);
43     }
44
45     return 0;
46 }
```

실습 2

- 사용자가 선택한 옵션에 걸맞는 동작을 수행할 수 있도록 해야겠죠?
- Switch-case를 이용해서 구현해봅시다
  - 0을 선택한 경우 : 프로그램 종료
  - 1을 선택한 경우: 새로운 학생을 등록
  - 2를 선택한 경우: 등록된 학생 출력

```c
while(option){
  printf("Choose option\n");
  printf("(0: Exit, 1: Student Registration, 2: Show Students)\n- ");
  scanf("%d", &option);

  switch(option){
    case 0:
      printf("Bye Bye~\n");
      break;
    case 1:
      if(!new_std(stds, &std_cnt))
        printf("Registration Failed\n");
      else
        printf("Registration Complete\n");
      break;
    case 2:
      show_stds(stds, std_cnt);
  }
}
```

실습 2

- 프로그램을 실행해서 정상적으로 동작하는지 확인해보세요!

실습 2

- 전체 코드

```c
#include <stdio.h>

#define MAX_STD 10

struct student{
    char name[10];
    char id[11];
};

```

실습 2

- 전체 코드

```
10 ▼ int new_std(struct student stds[], int* std_cnt){
11     if(*std_cnt == MAX_STD)
12       return 0;
13
14     printf("Name: ");
15     scanf("%s", stds[*std_cnt].name);
16
17     printf("ID: ");
18     scanf("%s", stds[*std_cnt].id);
19
20     (*std_cnt)++;
21     return 1;
22   }
23
```

실습 2

- 전체 코드

```c
void show_stds(struct student stds[], int std_cnt){
  printf("---------\n");
  for(int i = 0; i < std_cnt; i++){
    printf("No.%d\n", i+1);
    printf("Name: %s\n", stds[i].name);
    printf("ID: %s\n", stds[i].id);
  }
  printf("---------\n");
}

int main(){
  struct student stds[MAX_STD];
  int std_cnt = 0;
  int option = -1;

  while(option){
    printf("Choose option\n");
    printf("(0: Exit, 1: Student Registration, 2: Show Students)\n- ");
    scanf("%d", &option);
```

실습 2

- 전체 코드

```
41      printf("(0: Exit, 1: Student Registration, 2: Show Students)\n- ");
42      scanf("%d", &option);
43
44      switch(option){
45        case 0:
46          printf("Bye Bye~\n");
47          break;
48        case 1:
49          if(!new_std(stds, &std_cnt))
50            printf("Registration Failed\n");
51          else
52            printf("Registration Complete\n");
53          break;
54        case 2:
55          show_stds(stds, std_cnt);
56      }
57    }
58
59    return 0;
60  }
```

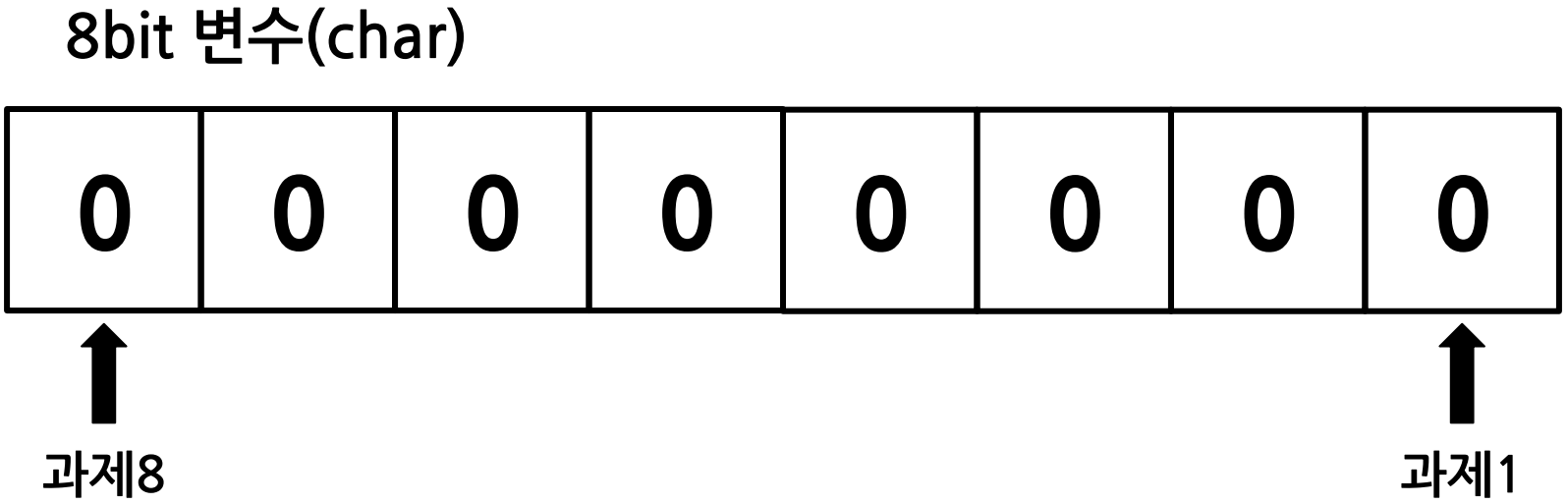
실습 3

실습 3 - 설명

- 학생들의 과제 제출 정보를 관리하는 기능을 추가해봅시다!
- 과제는 총 8개입니다
- 각 과제의 제출 여부만 다릅니다
  - 각 과제의 점수는 다루지 않습니다
- 연습을 위해 Bitwise operation을 사용해봅시다!

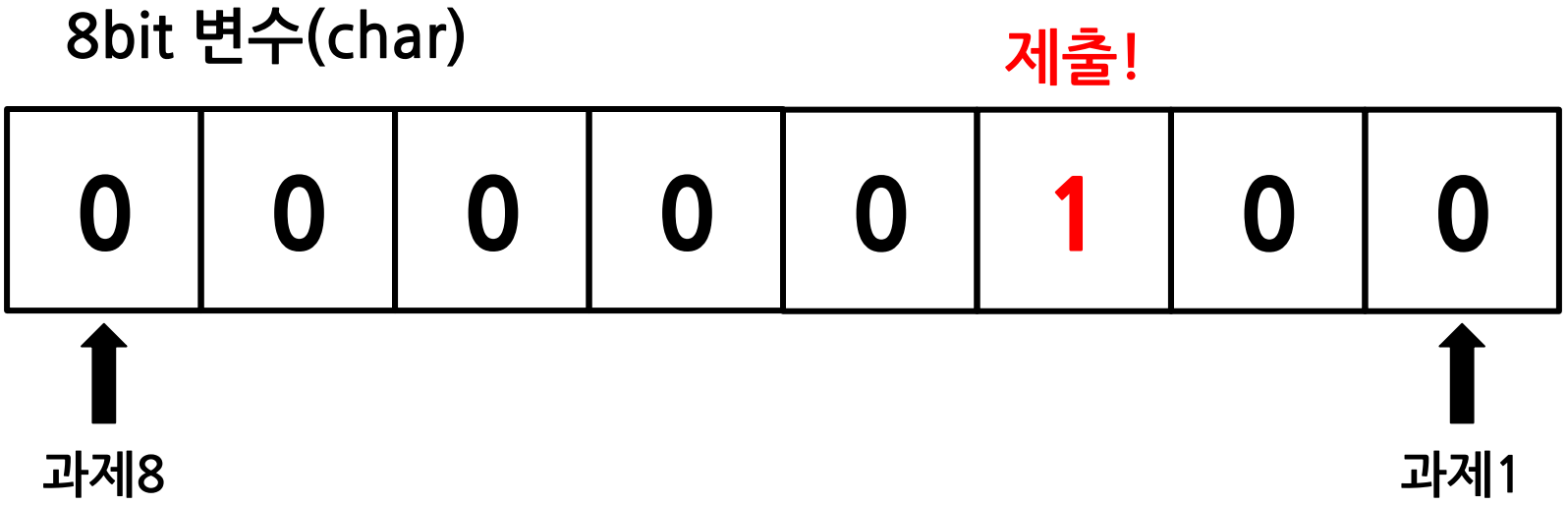
실습 3 - 설명

- 8개 과제의 제출 여부를 저장하는데는, 1Byte(8bit)면 충분합니다!



실습 3 - 설명

- 8개 과제의 제출 여부를 저장하는데는, 1Byte(8bit)면 충분합니다!



실습 3 - 설명

- 8개 과제의 제출 여부를 저장하는데는, 1Byte(8bit)면 충분합니다!

실습 3 - 설명

- 학생의 정보를 저장하는 struct에 submitted 변수를 추가합니다
  - char 변수로 만들어보세요!
- 만든 submitted 변수를 0으로 초기화 해야합니다
  - new_std 함수에서 새로운 학생을 등록하면서 같이 하면 되겠죠?

실습 3 - 설명

- 코드 예시

```c
struct student{
    char name[10];
    char id[11];
    char submitted;
};
```

```c
int new_std(struct student stds[], int* std_cnt){
    if(*std_cnt == MAX_STD)
        return 0;

    printf("Name: ");
    scanf("%s", stds[*std_cnt].name);

    printf("ID: ");
    scanf("%s", stds[*std_cnt].id);

    stds[*std_cnt].submitted = 0;

    (*std_cnt)++;
    return 1;
}
```

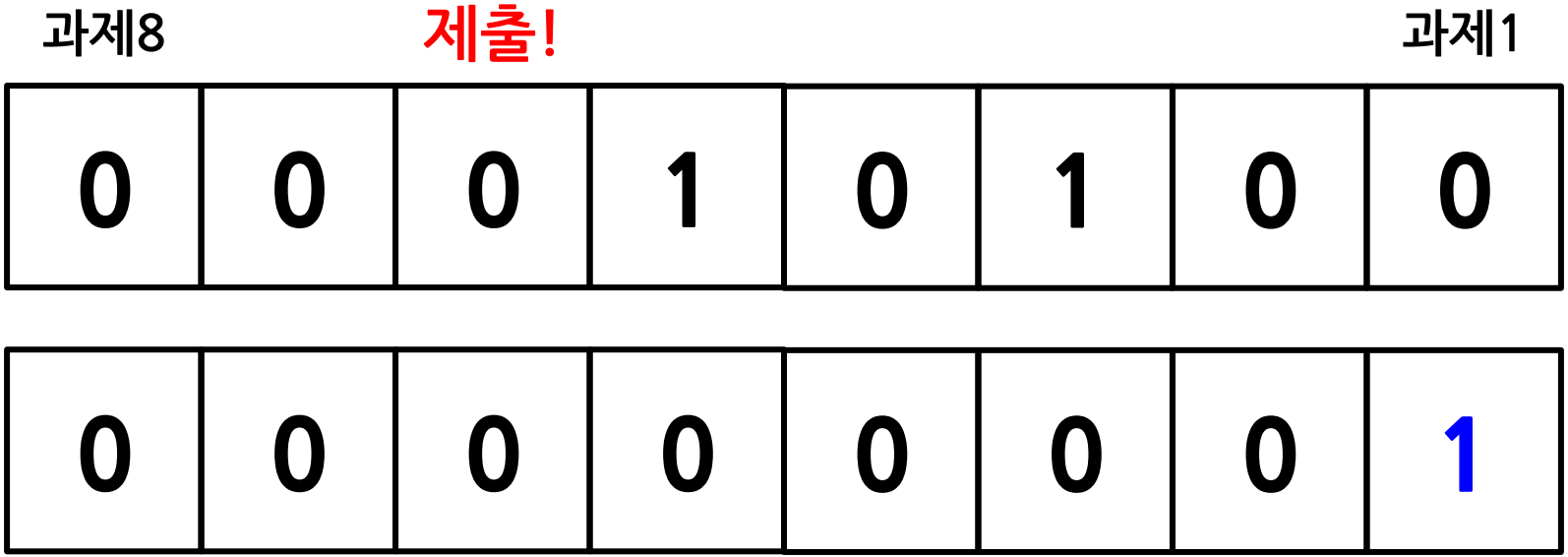
실습 3 - 설명

- 함수를 2개로 구성할 예정입니다
  - 다수의 제출 과제를 처리하는 함수(hand_in_multi)
  - 제출한 하나의 과제를 submitted 변수에 저장하는 함수(hand_in_once)
- hand_in_multi 함수는 hand_in_once 함수를 활용합니다!
  - 제출한 과제들 각각을 submitted에 저장하는 것을 hand_in_once을 활용하여 수행합니다

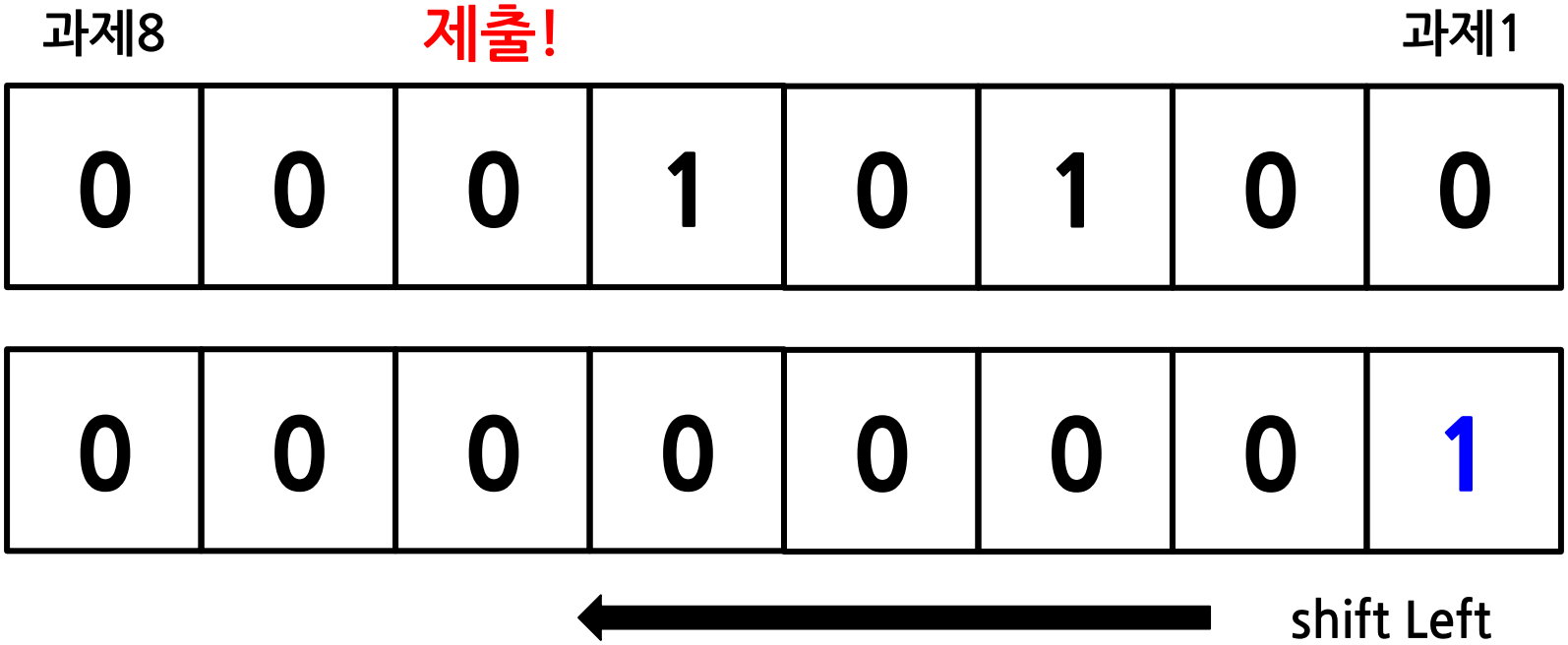
실습 3 - 설명

- hand_in_once 함수를 만들어 봅시다!
  - 원하는 bit를 set 하는 것은, bit masking을 이용하면 수월합니다

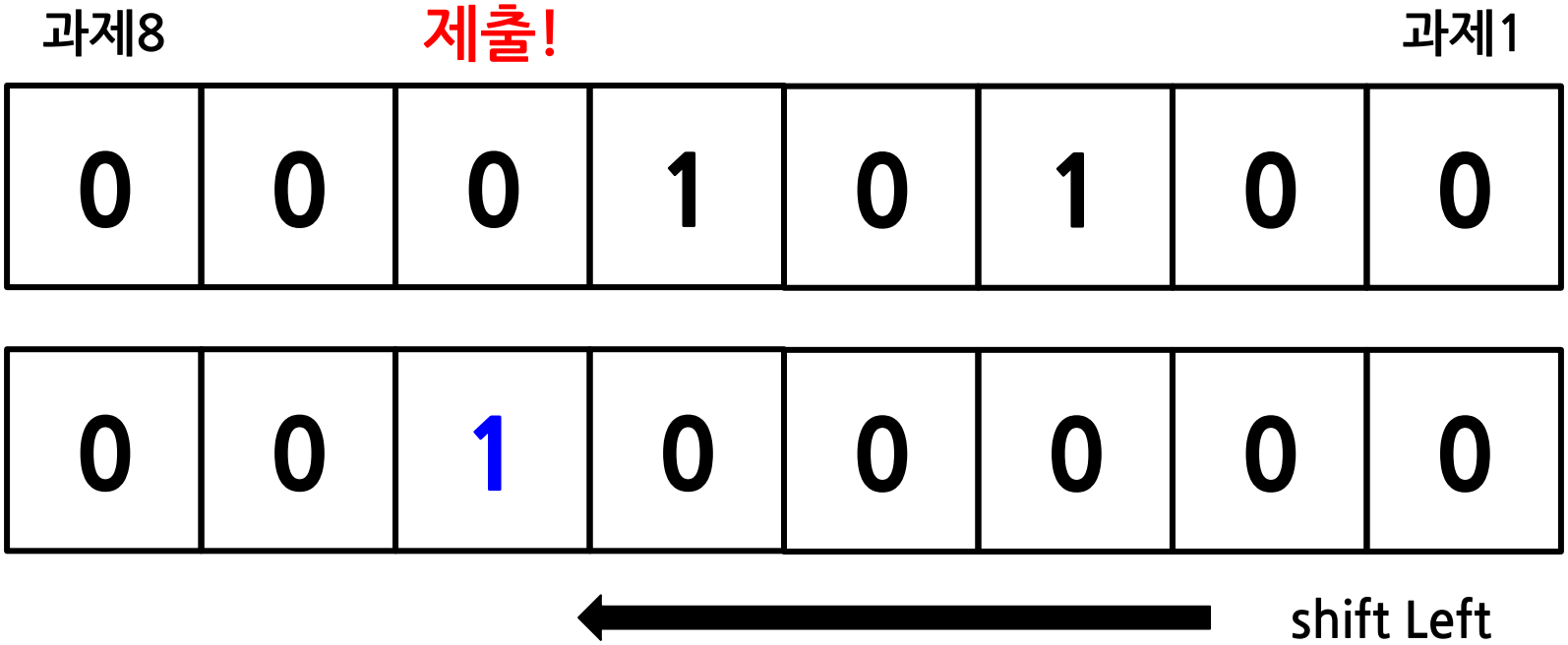
실습 3 - 설명



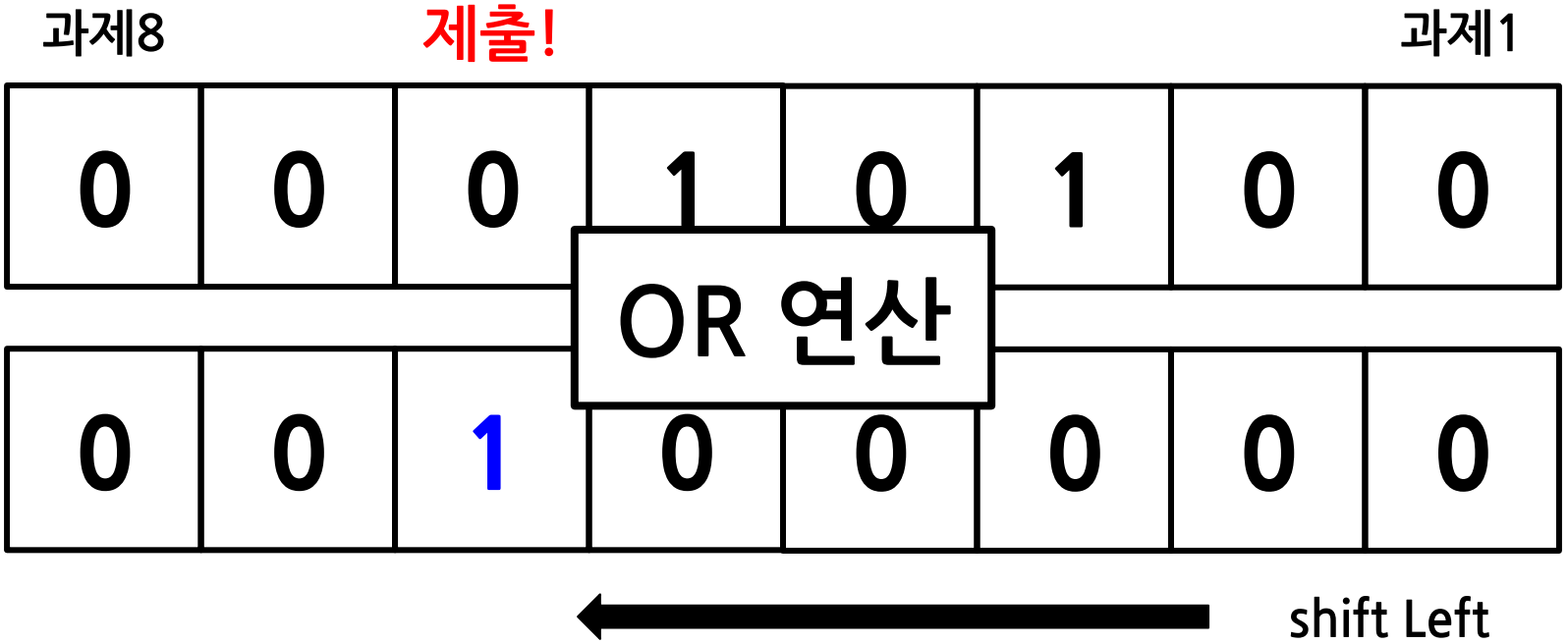
실습 3 - 설명



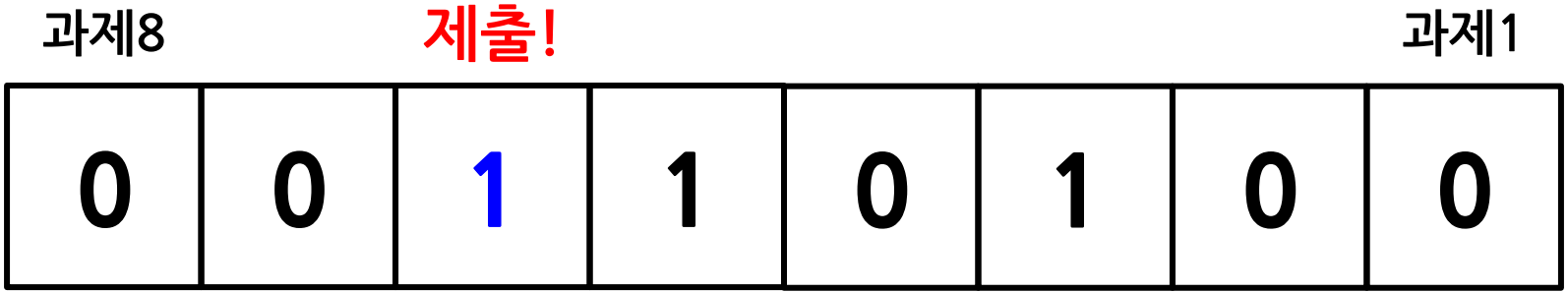
실습 3 - 설명



실습 3 - 설명



실습 3 - 설명



실습 3 - 설명

- 코드 예시

```
37 ▼ void hand_in_once(char *submitted, int num){
38       *submitted |= 1 << (num-1);
39   }
```

실습 3 - 설명

- hand_in_multi 함수를 만들어봅시다!
  - 사용자에게 제출된 과제 정보를 한번에 받아서 submitted에 저장하는 함수입니다
- 입력은 다음과 같이 받습니다
  - [제출된 과제 개수] [과제 번호들]
  - 예) 3 1 2 3
  - 예) 5 3 5 2 7 8
- 함수에서 직접 입력을 받아 정보를 저장할 수 있도록 구현해보세요
  - 앞에서 언급했듯이, 미리 만들어둔 hand_in_once를 활용해보세요

실습 3 - 설명

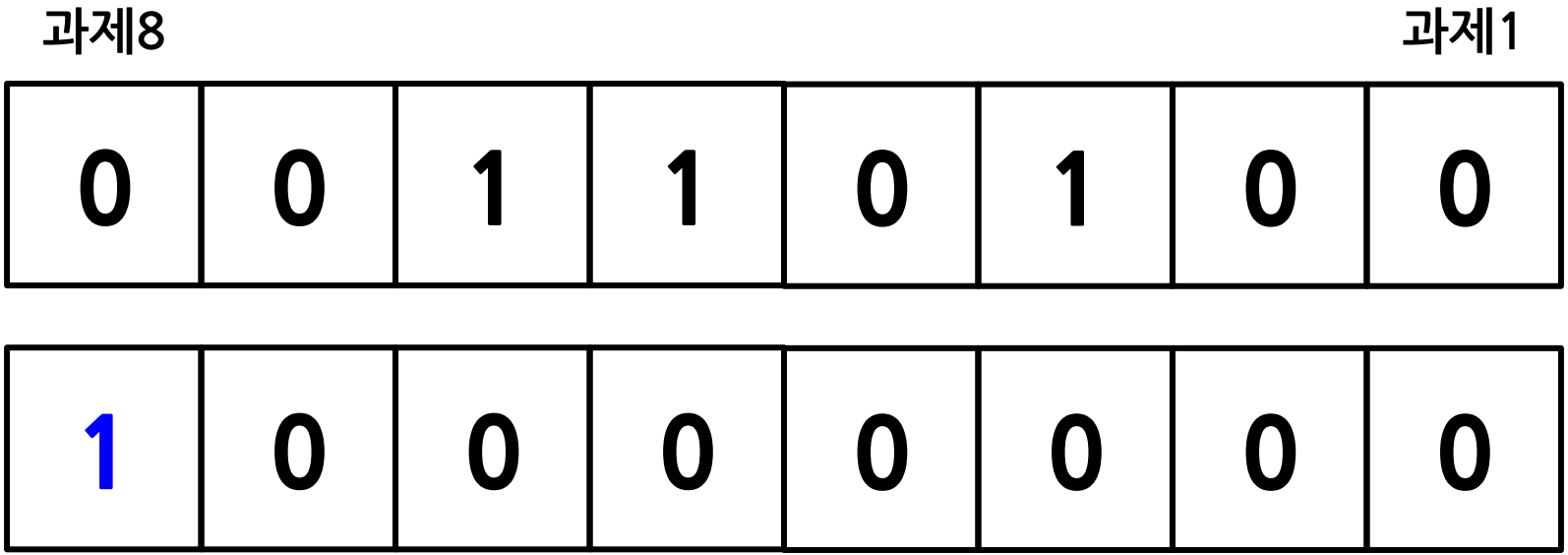
- 코드 예시

```
31 ▼ void hand_in_multi(struct student stds[], int std_num){
32       int input_num, assignment;
33
34       printf("Insert input: ");
35       scanf("%d", &input_num);
36 ▼     for(int i = 0; i < input_num; i++){
37           scanf("%d", &assignment);
38           hand_in_once(&stds[std_num].submitted, assignment);
39       }
40   }
```

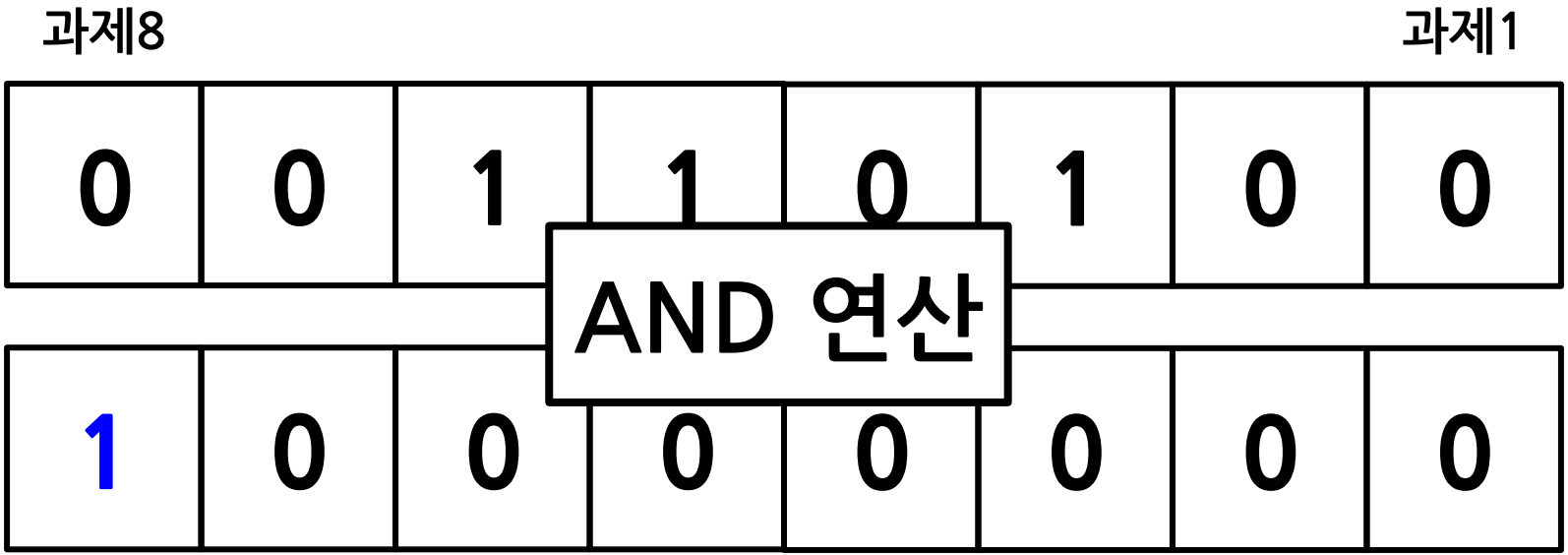
실습 3 - 설명

- 이제 제출된 과제가 잘 반영되었는지 확인해야겠죠?
- submitted 변수를 2진수로 출력하는 함수(show_submit)를 만들어봅시다
  - 교수님의 강의 자료에 있는 방법을 활용합니다!

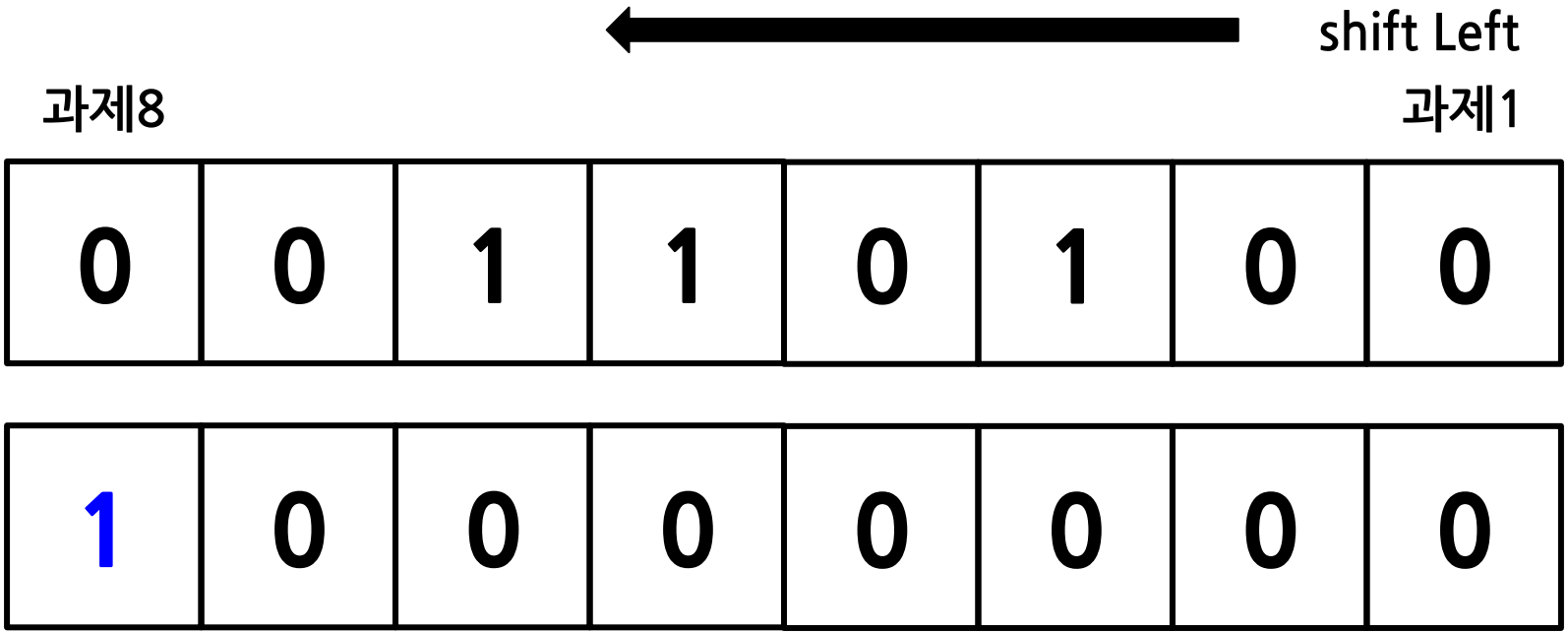
실습 3 - 설명



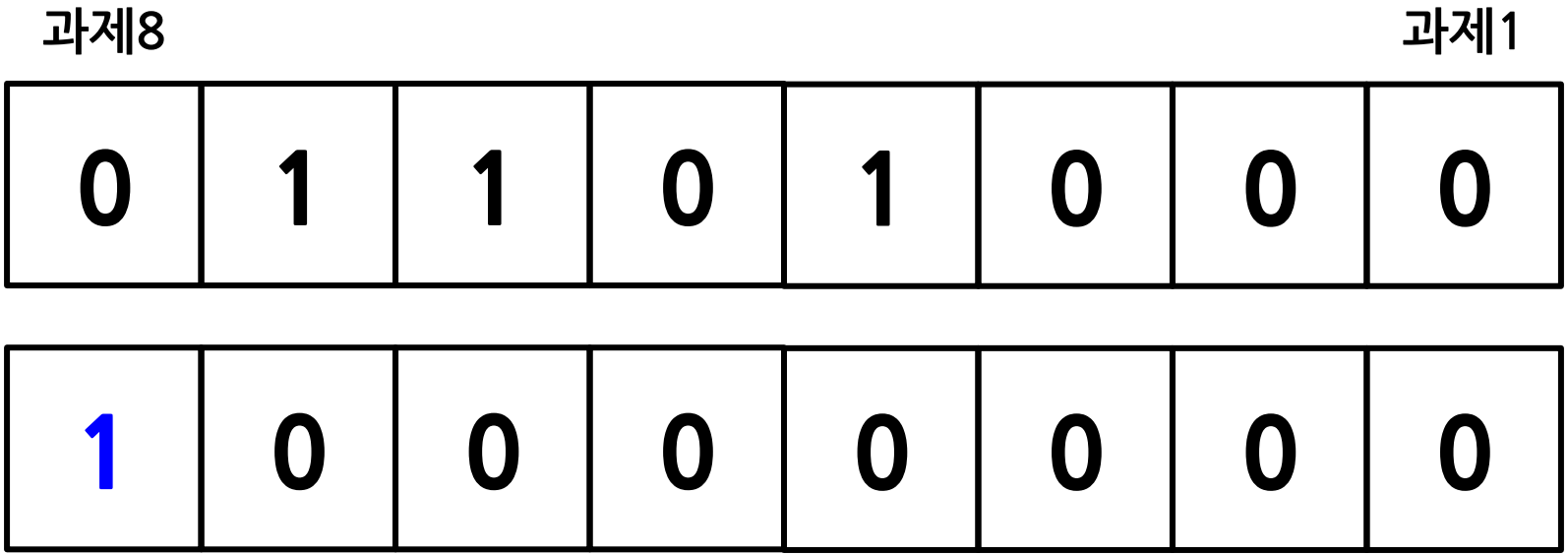
실습 3 - 설명



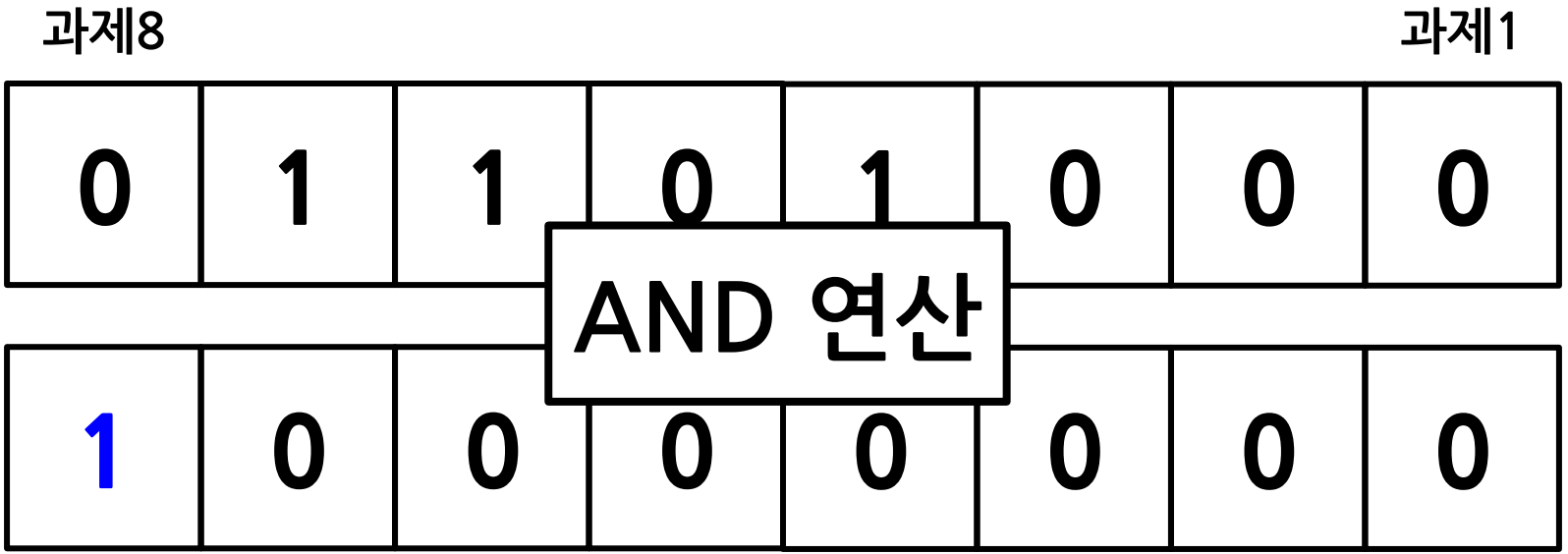
실습 3 - 설명



실습 3 - 설명



실습 3 - 설명



실습 3 - 설명

- 코드 예시

```
51 ▼ void show_submit(char submitted){
52     char mask = 1;
53     mask <<= 7;
54 ▼   for(int i = 0; i < 8; i++){
55       putchar(submitted & mask ? '1' : '0');
56       submitted <<= 1;
57     }
58     putchar('\n');
59   }
```

실습 3

- 마찬가지로 모든 함수들을 활용해서 main 함수에 과제 관리 기능을 추가해봅시다!
  - option 3번에 Submit Assignment 항목을 추가합니다
  - 사용자가 3번을 선택한다면, 과제를 제출할 학생을 입력 받습니다
  - hand_in_mulit 함수를 이용해서 과제 등록을 수행합니다
- 과제 제출 정보는 option 2번(Show Students)에서 함께 보여줍니다
  - show_stds 함수를 수정해보세요!

실습 3

- 실행 결과

```
> make -s
> ./main
Choose option
(0: Exit, 1: Student Registration, 2: Show Students, 3: Submit Assignment)
- 1
Name: jslee
ID: 2099-99999
Registration Complete
Choose option
(0: Exit, 1: Student Registration, 2: Show Students, 3: Submit Assignment)
- 2
---------
No.1
Name: jslee
ID: 2099-99999
Submitted: 00000000
---------
```

실습 3

- 실행 결과

```
Choose option
(0: Exit, 1: Student Registration, 2: Show Students, 3: Submit Assignment)
- 3
Student No: 1
Insert input: 5 2 6 7 4 1
Choose option
(0: Exit, 1: Student Registration, 2: Show Students, 3: Submit Assignment)
- 2
---------
No.1
Name: jslee
ID: 2099-99999
Submitted: 01101011
---------
Choose option
(0: Exit, 1: Student Registration, 2: Show Students, 3: Submit Assignment)
- 0
Bye Bye~
```

실습 3

- 코드 예시

```
64 ▼ int main(){
65     struct student stds[MAX_STD];
66     int std_cnt = 0;
67     int option = -1;
68     int std;
69
70 ▼   while(option){
71       printf("Choose option\n");
72       printf("(0: Exit, 1: Student Registration, 2: Show Students, 3:
   Submit Assignment)\n- ");
73       scanf("%d", &option);
74
```

실습 3

- 코드 예시

```
75 ▼    switch(option){
76        case 0:
77          printf("Bye Bye~\n");
78          break;
79        case 1:
80          if(!new_std(stds, &std_cnt))
81            printf("Registration Failed\n");
82          else
83            printf("Registration Complete\n");
84          break;
85        case 2:
86          show_stds(stds, std_cnt);
87          break;
88        case 3:
89          printf("Student No: ");
90          scanf("%d", &std);
91          hand_in_multi(stds, std-1);
92          break;
93      }
94    }
```

실습 3

- 코드 예시

```
52 void show_stds(struct student stds[], int std_cnt){
53     printf("---------\n");
54     for(int i = 0; i < std_cnt; i++){
55         printf("No.%d\n", i+1);
56         printf("Name: %s\n", stds[i].name);
57         printf("ID: %s\n", stds[i].id);
58         printf("Submitted: ");
59         show_submit(stds[i].submitted);
60     }
61     printf("---------\n");
62 }
```

실습 3

- 잘 동작하는지 확인해보세요!

실습 4

실습 4 - **제출**

- 마지막으로 GPA를 계산하고 저장하는 기능을 추가해봅시다!
  - 연습을 위해서 enum을 이용해봅시다
- GPA의 기준은 다음과 같습니다
  - 과제 제출의 개수를 기준으로 삼습니다
  - 8개 제출 : A(4)
  - 7개 제출 : B(3)
  - 4~6개 제출 : C(2)
  - 1~3개 제출 : D(1)
  - 0개 제출 : F(0)

실습 4 - 설명

- 먼저 enum을 이용해서 GPA를 만들어봅시다
- 각 학생들 마다 GPA 정보를 저장하기 위해, student 구조체 안에 GPA 변수도 추가합니다

```
5  enum GPA {
6      F,
7      D,
8      C,
9      B,
10     A
11 };
```

```
13 struct student{
14     char name[10];
15     char id[11];
16     char submitted;
17     enum GPA gpa;
18 };
```

실습 4 - 설명

- GPA를 계산하여 저장하는 함수를 만들어봅시다
- 먼저, 제출된 과제의 개수를 계산하는 함수(submit_cnt)가 필요합니다
  - 마찬가지로 bit 연산을 사용해봅시다!
  - submitted 변수에 저장된 1의 개수를 count 해서 return 합니다

실습 4 - 설명

- 코드 예시

```
61 ▼ int submit_cnt(char submitted){
62       int cnt = 0;
63 ▼     for(int i = 0 ; i < 8; i++){
64         if(submitted & 1)
65           cnt++;
66         submitted >>= 1;
67       }
68       return cnt;
69   }
```

실습 4 - 설명

- 원하는 학생의 GPA를 계산하여 저장하는 함수(calc_GPA)를 만들어봅시다
  - calc_GPA는 submit_cnt 함수를 이용하여 해당 학생이 제출한 과제의 개수를 알아냅니다
  - 제출한 과제의 개수를 기준으로 GPA를 계산하여 gpa 변수에 저장합니다

실습 4 - 설명

- 코드 예시

```
71 ▼ void calc_GPA(struct student *std){
72       int cnt = submit_cnt(std->submitted);
73
74       if(cnt == 8)
75           std->gpa = A;
76       else if(cnt == 7)
77           std->gpa = B;
78       else if(cnt >= 4)
79           std->gpa = C;
80       else if(cnt >= 1)
81           std->gpa = D;
82       else
83           std->gpa = F;
84   }
```

실습 4 - **제출**

- 만든 함수들을 바탕으로 학생들의 GPA를 계산하고 저장하는 기능을 완성해봅시다
  - 학생을 새로 등록할 때, GPA를 F로 초기화합니다
  - show_stds함수가 GPA 정보도 출력할 수 있도록 변경해보세요
- 학생들의 GPA는 과제가 제출될 때 마다 자동으로 갱신되도록 합니다
  - hand_in_multi 함수의 마지막 부분에서, calc_GPA를 활용하면 되겠죠?

실습 4

- 실행 결과

```
make -s
./main
Choose option
(0: Exit, 1: Student Registration, 2: Show Students, 3: Submit Assignment)
- 1
Name: jslee
ID: 2099-99999
Registration Complete
Choose option
(0: Exit, 1: Student Registration, 2: Show Students, 3: Submit Assignment)
- 2
---------
No.1
Name: jslee
ID: 2099-99999
Submitted: 00000000
GPA: 0
---------
```

실습 4

- 실행 결과

```
Choose option
(0: Exit, 1: Student Registration, 2: Show Students, 3: Submit Assignment)
- 3
Student No: 1
Insert input: 2 1 2
Choose option
(0: Exit, 1: Student Registration, 2: Show Students, 3: Submit Assignment)
- 2
---------
No.1
Name: jslee
ID: 2099-99999
Submitted: 00000011
GPA: 1
---------
```

실습 4

- 실행 결과

```
Choose option
(0: Exit, 1: Student Registration, 2: Show Students, 3: Submit Assignment)
- 3
Student No: 1
Insert input: 6 3 4 5 6 7 8
Choose option
(0: Exit, 1: Student Registration, 2: Show Students, 3: Submit Assignment)
- 2
---------
No.1
Name: jslee
ID: 2099-99999
Submitted: 11111111
GPA: 4
---------
Choose option
(0: Exit, 1: Student Registration, 2: Show Students, 3: Submit Assignment)
- 0
Bye Bye~
```

실습 4

- 코드 예시

```
20 ▼ int new_std(struct student stds[], int* std_cnt){
21     if(*std_cnt == MAX_STD)
22       return 0;
23
24     printf("Name: ");
25     scanf("%s", stds[*std_cnt].name);
26
27     printf("ID: ");
28     scanf("%s", stds[*std_cnt].id);
29
30     stds[*std_cnt].submitted = 0;
31
32     stds[*std_cnt].gpa = F;
33
34     (*std_cnt)++;
35     return 1;
36 }
```

실습 4

- 코드 예시

```
67 ▼ void hand_in_multi(struct student stds[], int std_num){
68     int input_num, assignment;
69
70     printf("Insert input: ");
71     scanf("%d", &input_num);
72 ▼   for(int i = 0; i < input_num; i++){
73         scanf("%d", &assignment);
74         hand_in_once(&stds[std_num].submitted, assignment);
75     }
76
77     calc_GPA(&stds[std_num]);
78 }
```

실습 4

- 코드 예시

```
90  void show_stds(struct student stds[], int std_cnt){
91      printf("---------\n");
92      for(int i = 0; i < std_cnt; i++){
93          printf("No.%d\n", i+1);
94          printf("Name: %s\n", stds[i].name);
95          printf("ID: %s\n", stds[i].id);
96          printf("Submitted: ");
97          show_submit(stds[i].submitted);
98          printf("GPA: %d\n", stds[i].gpa);
99      }
100     printf("---------\n");
101 }
```

Appendix

Appendix

```c
#include <stdio.h>

#define MAX_STD 10

enum GPA {
    F,
    D,
    C,
    B,
    A
};

struct student{
    char name[10];
    char id[11];
    char submitted;
    enum GPA gpa;
};
```

Appendix

```c
int new_std(struct student stds[], int* std_cnt){
  if(*std_cnt == MAX_STD)
    return 0;

  printf("Name: ");
  scanf("%s", stds[*std_cnt].name);

  printf("ID: ");
  scanf("%s", stds[*std_cnt].id);

  stds[*std_cnt].submitted = 0;

  stds[*std_cnt].gpa = F;

  (*std_cnt)++;
  return 1;
}
```

Appendix

```
38 ▼ int submit_cnt(char submitted){
39       int cnt = 0;
40 ▼     for(int i = 0 ; i < 8; i++){
41         if(submitted & 1)
42           cnt++;
43         submitted >>= 1;
44       }
45       return cnt;
46   }
```

Appendix

```
48 ▼ void calc_GPA(struct student *std){
49     int cnt = submit_cnt(std->submitted);
50
51     if(cnt == 8)
52       std->gpa = A;
53     else if(cnt == 7)
54       std->gpa = B;
55     else if(cnt >= 4)
56       std->gpa = C;
57     else if(cnt >= 1)
58       std->gpa = D;
59     else
60       std->gpa = F;
61   }
```

Appendix

```c
void hand_in_once(char *submitted, int num){
  *submitted |= 1 << (num-1);
}

void hand_in_multi(struct student stds[], int std_num){
  int input_num, assignment;

  printf("Insert input: ");
  scanf("%d", &input_num);
  for(int i = 0; i < input_num; i++){
    scanf("%d", &assignment);
    hand_in_once(&stds[std_num].submitted, assignment);
  }

  calc_GPA(&stds[std_num]);
}
```

Appendix

```
80 ▼ void show_submit(char submitted){
81       char mask = 1;
82       mask <<= 7;
83 ▼     for(int i = 0; i < 8; i++){
84           putchar(submitted & mask ? '1' : '0');
85           submitted <<= 1;
86       }
87       putchar('\n');
88   }
```

Appendix

```
90 ▼ double calc_GPA_avg(struct student stds[], int std_cnt){
91     if(std_cnt == 0)
92       return 0;
93
94     double sum = 0;
95
96     for(int i = 0; i < std_cnt; i++)
97       sum += stds[i].gpa;
98
99     return sum / std_cnt;
100  }
```

Appendix

```
102 ▼ void show_stds(struct student stds[], int std_cnt){
103       printf("---------\n");
104 ▼     for(int i = 0; i < std_cnt; i++){
105           printf("No.%d\n", i+1);
106           printf("Name: %s\n", stds[i].name);
107           printf("ID: %s\n", stds[i].id);
108           printf("Submitted: ");
109           show_submit(stds[i].submitted);
110           printf("GPA: %d\n", stds[i].gpa);
111       }
112       printf("---------\n");
113   }
```

Appendix

```
115 int main(){
116     struct student stds[MAX_STD];
117     int std_cnt = 0;
118     int option = -1;
119     int std;
120 
121     while(option){
122         printf("Choose option\n");
123         printf("(0: Exit, 1: Student Registration, 2: Show Students,
    3: Submit Assignment, 4: GPA Average)\n- ");
124         scanf("%d", &option);
```

Appendix

```
126 ▼   switch(option){
127       case 0:
128         printf("Bye Bye~\n");
129         break;
130       case 1:
131         if(!new_std(stds, &std_cnt))
132           printf("Registration Failed\n");
133         else
134           printf("Registration Complete\n");
135         break;
136       case 2:
137         show_stds(stds, std_cnt);
138         break;
```

Appendix

```
139         case 3:
140           printf("Student No: ");
141           scanf("%d", &std);
142           hand_in_multi(stds, std-1);
143           break;
144         case 4:
145           printf("GPA Average: %0.2lf\n", calc_GPA_avg(stds, std_cnt));
146           break;
147       }
148     }
149
150     return 0;
151   }
```

과제 1

과제1

- 원하는 석차의 학생을 찾아내는 프로그램을 만들어 봅세요
- 각 학생 마다 다음의 정보가 주어집니다
  - 학번 - 학생마다 고유, 자연수로만 이루어져 있음
  - 점수 - 자연수로만 이루어져 있음, 동점자 존재
- 학생의 석차는 다음의 규칙으로 결정됩니다
  - 점수가 높은 순서대로 석차를 매깁니다
  - 동점자의 경우, 학번이 작은(앞번호) 가 더 높은 석차를 갖습니다

과제1

- 입력은 다음과 같이 주어집니다
  - [학생 수] [학생 1의 학번] [학생 1의 점수] [학생 2의 학번] [학생 2의 점수] …
  - [찾아보고 싶은 학생의 석차]
- 출력하셔야 하는 결과는 다음과 같습니다
  - [해당 석차 학생의 학번] [해당 석차 학생의 점수]

```
> make -s
> ./main
5 1 60 2 60 3 78 4 90 5 100
4
1 60
>
```

```
> make -s
> ./main
8 1 60 2 60 3 60 4 60 5 60 6 60 7 60 8 60
3
3 60
>
```

과제1

- 문제는 다음의 가정을 바탕으로 합니다
  - 최대 학생수는 100명입니다
  - 잘못된 입력은 없습니다
  - 학생의 학번과 점수의 범위는 int형이 표현할 수 있는 범위 안에 있습니다
- 조건을 만족하는 프로그램을 작성해보세요!

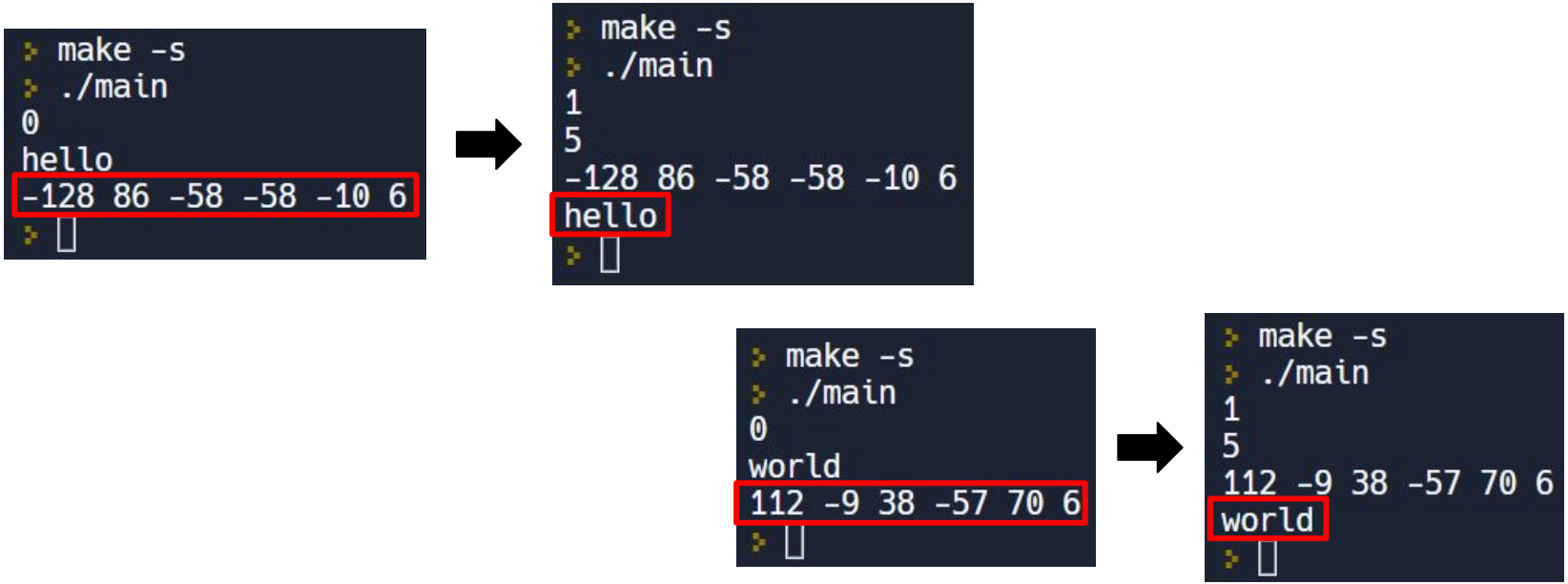
과제1 - 평가 기준

- 총 5점
- 5개의 test case 수행
- 1개 실패 마다 1점씩 감점

과제 2

과제2

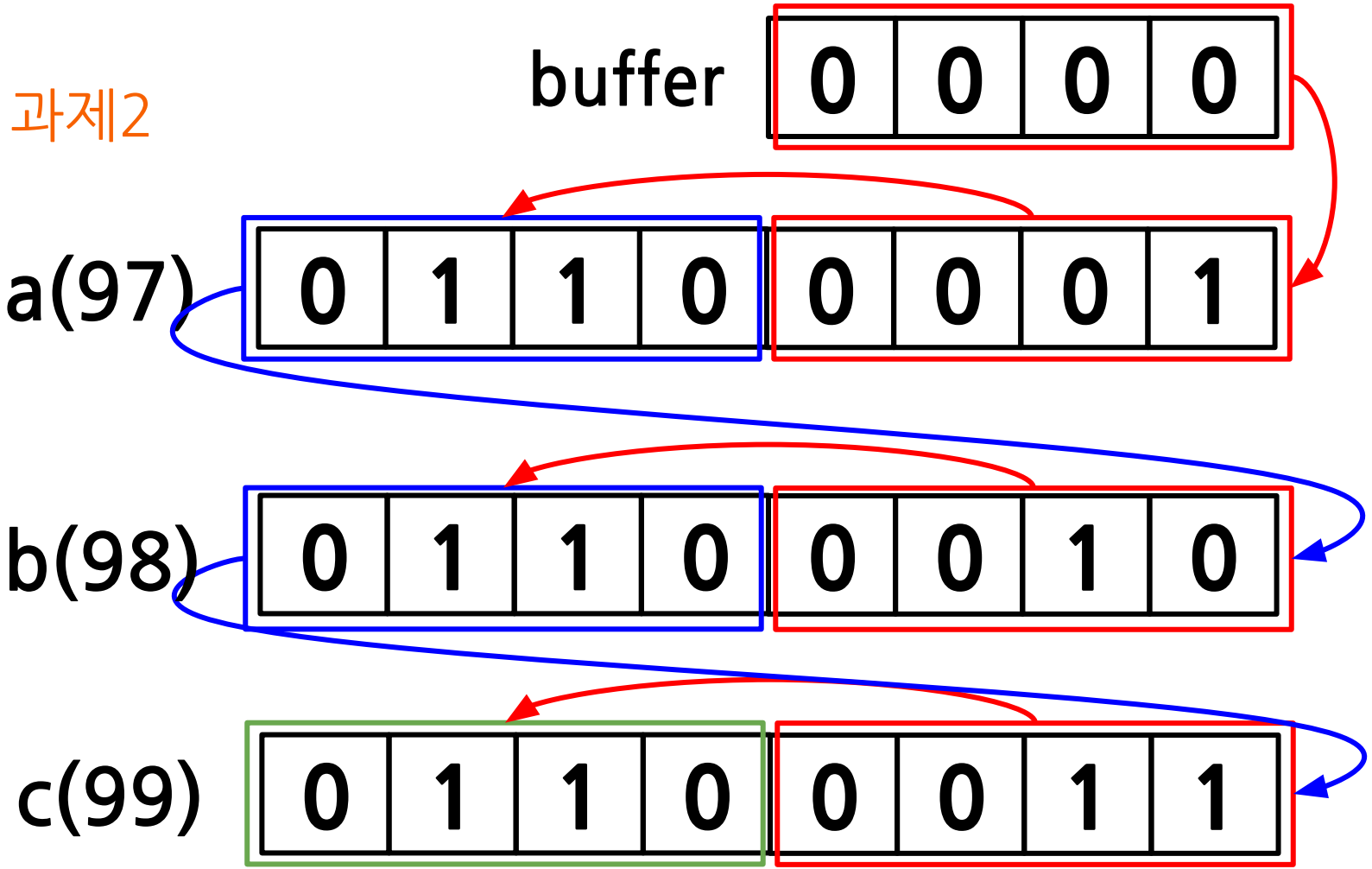
- Bitwise operation을 이용해서 암호/복호 프로그램을 만들어봅시다!



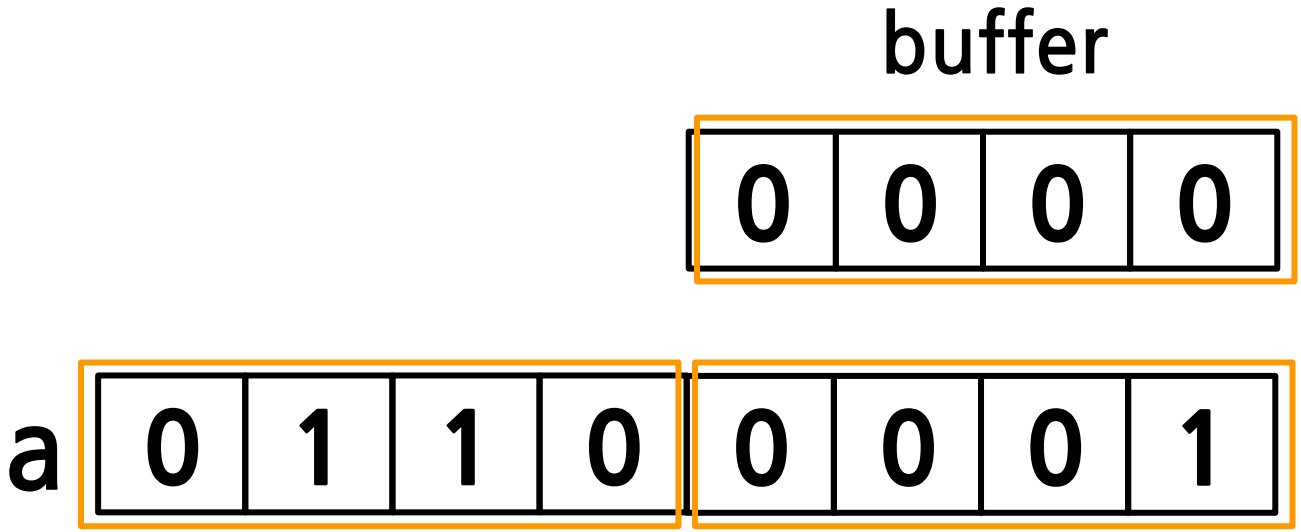
과제2

- 암호화 알고리즘은 다음과 같습니다
  - “abc”를 암호화 하는 과정

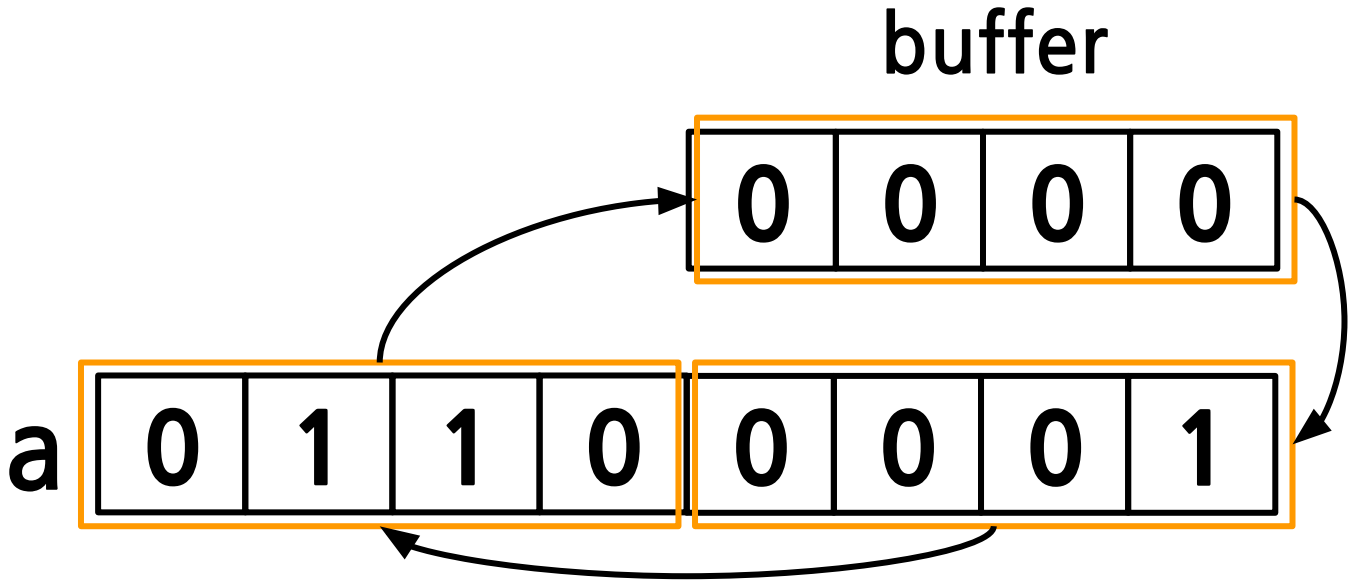
과제2



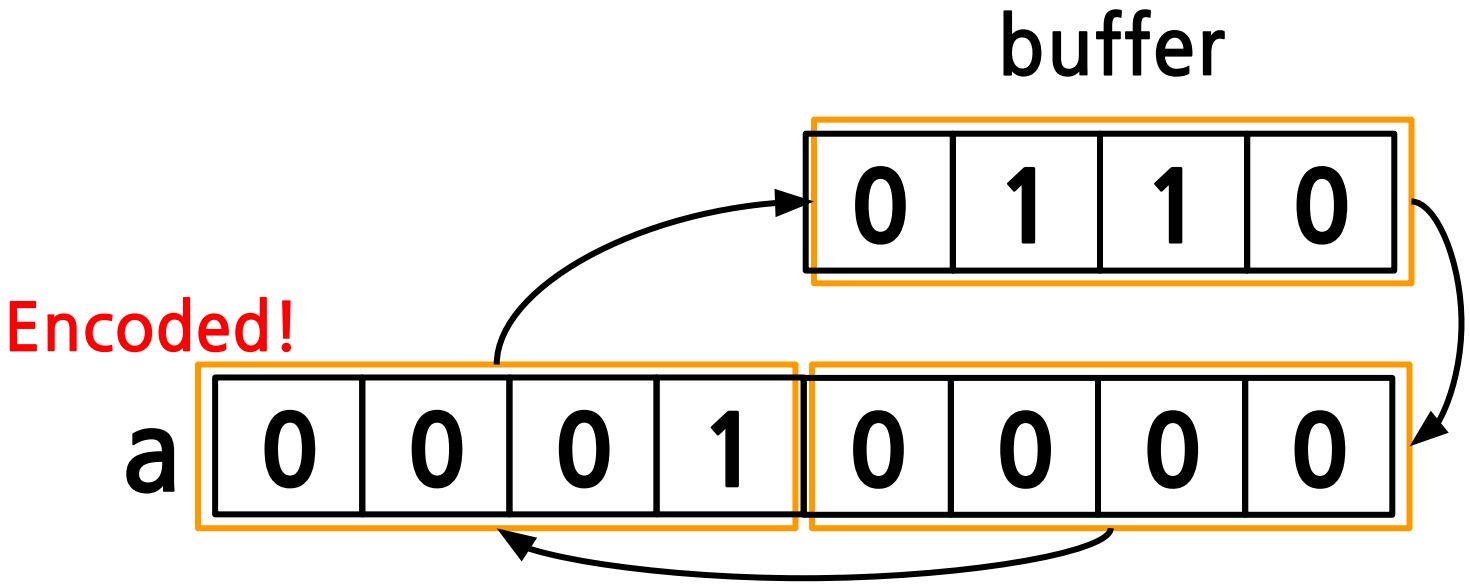
과제2



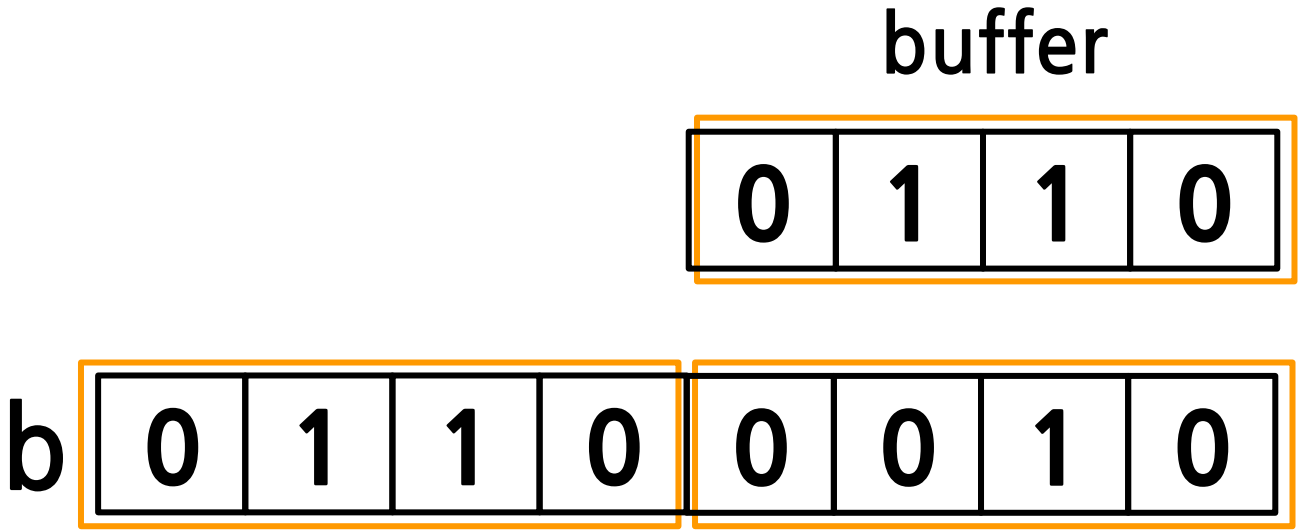
과제2



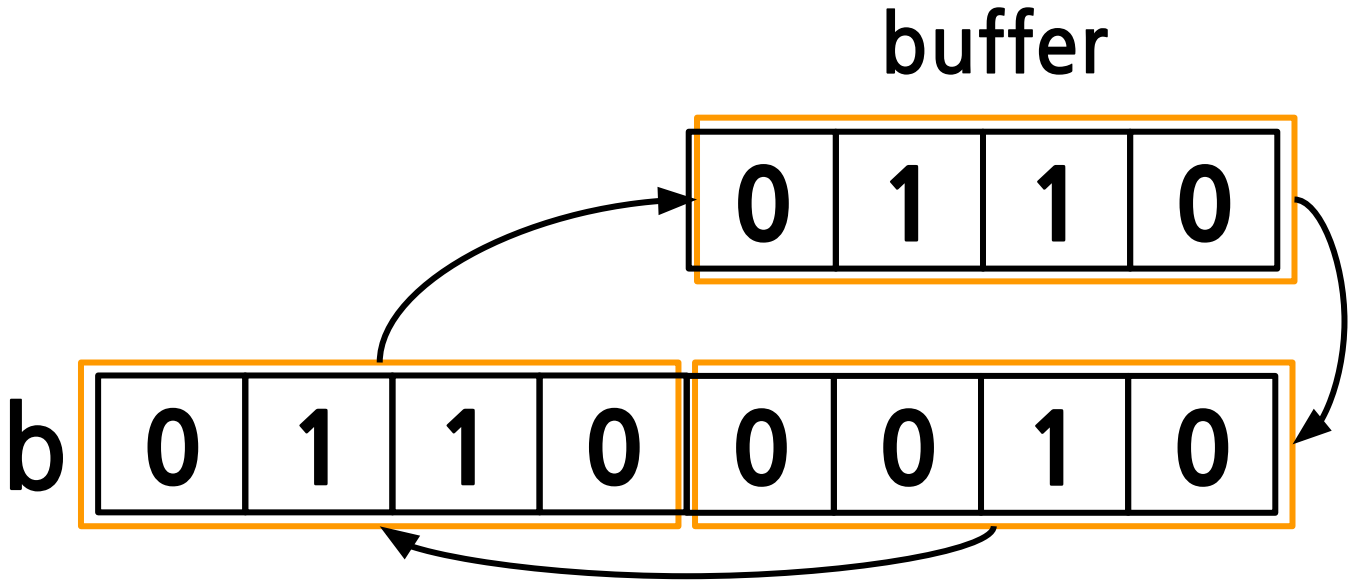
과제2



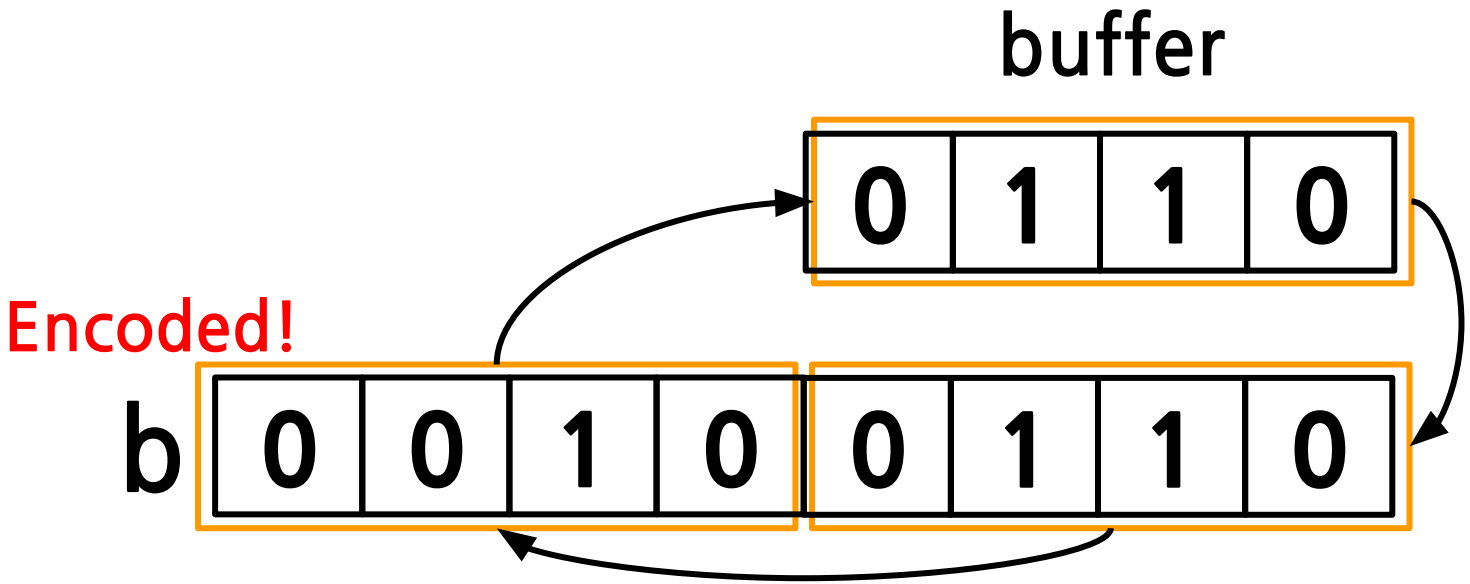
과제2



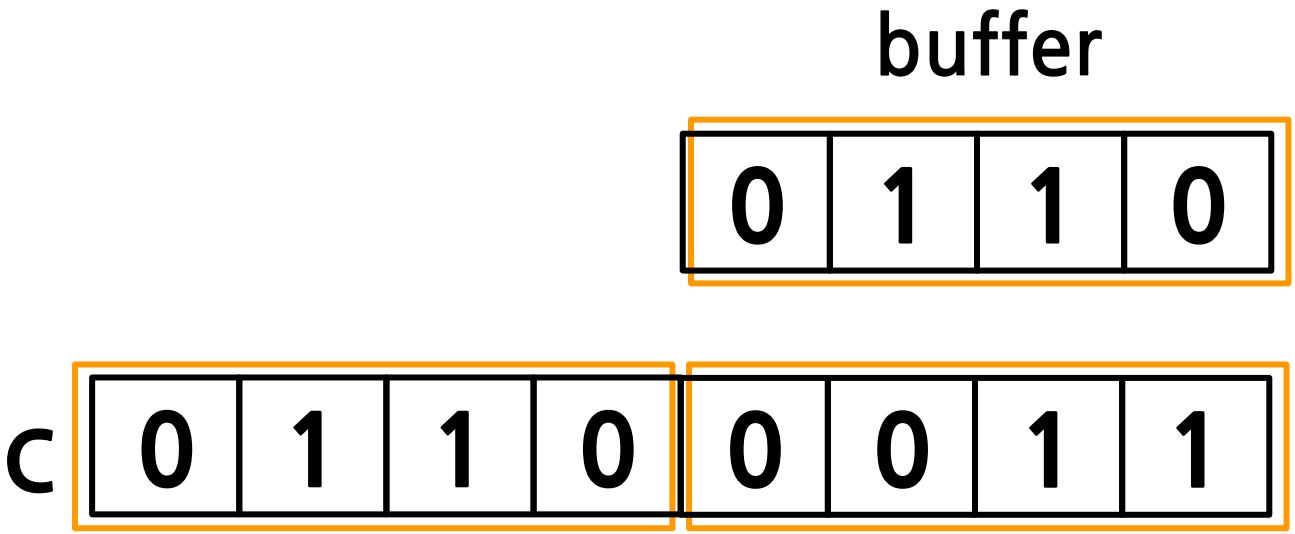
과제2



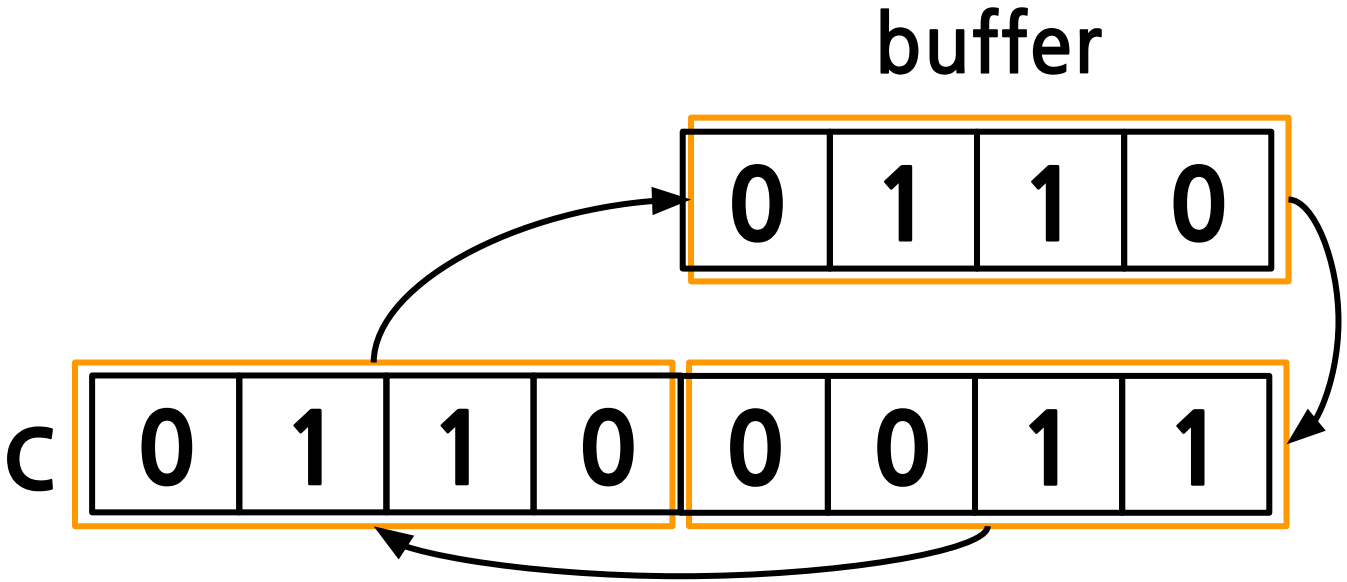
과제2



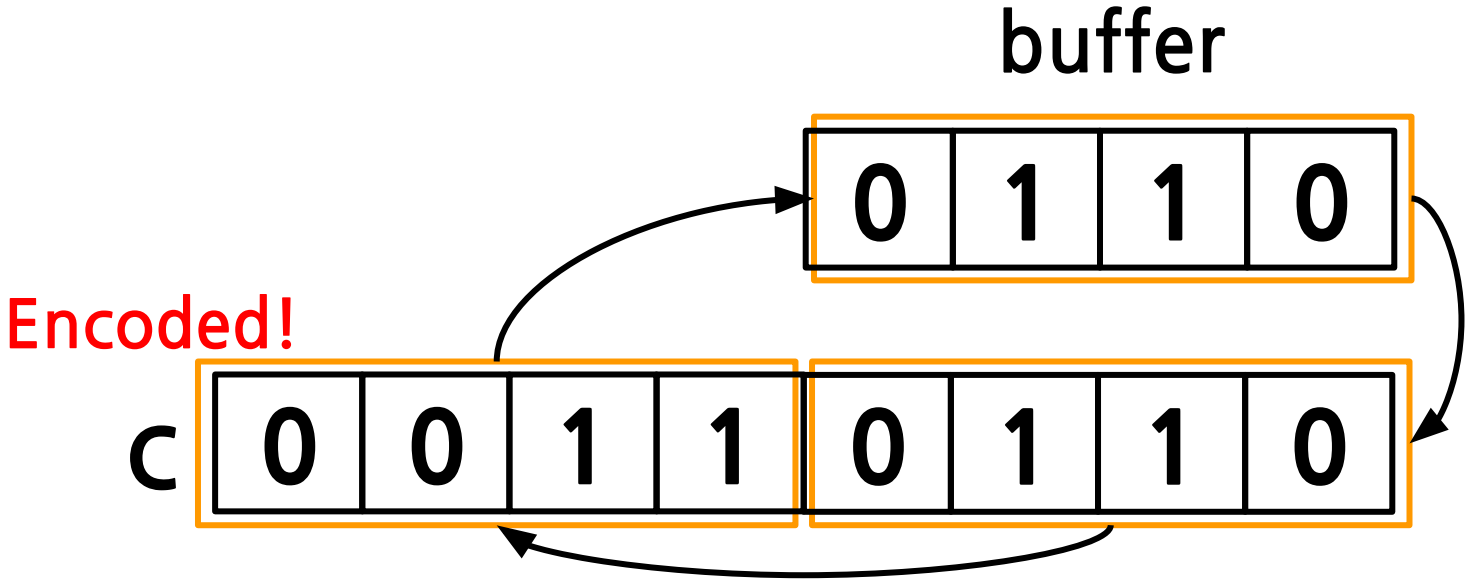
과제2



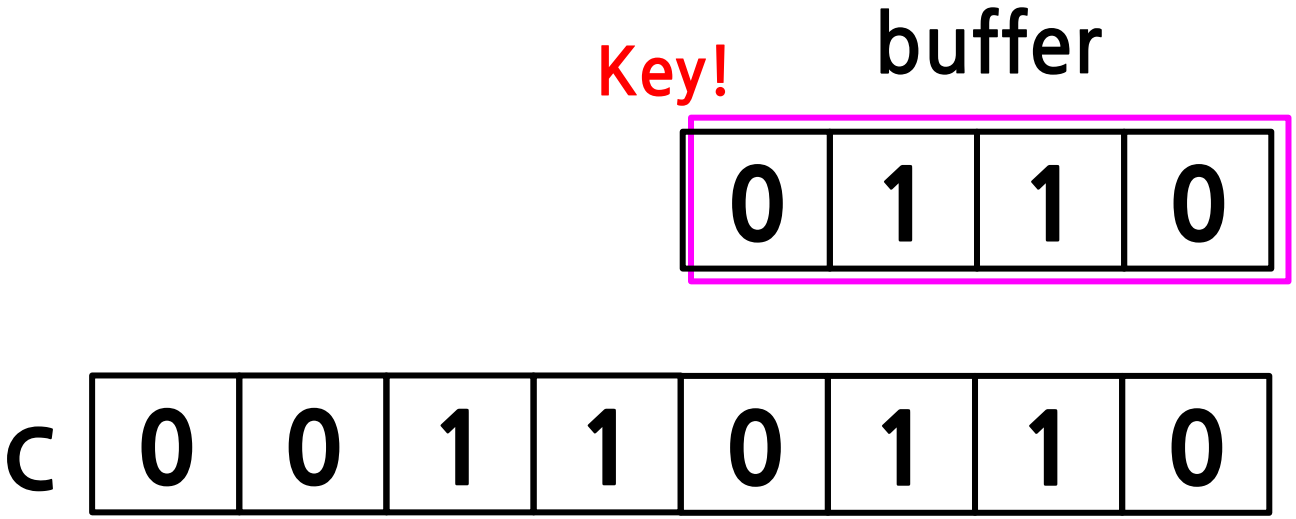
과제2



과제2



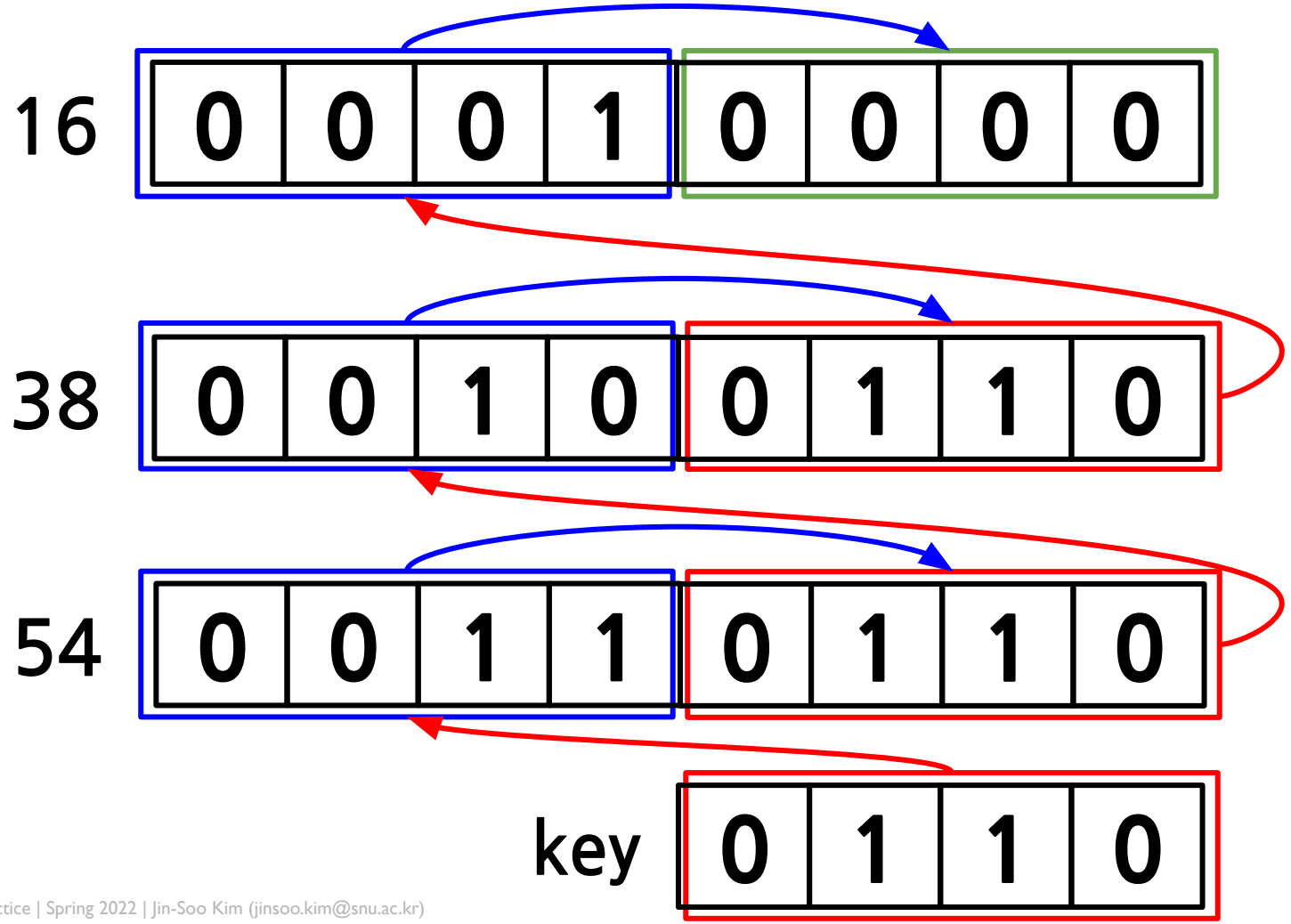
과제2



과제2

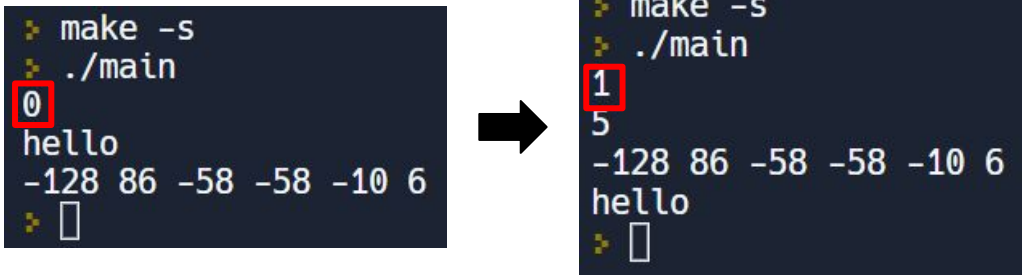
- 복호화는 암호화의 반대 과정 입니다!

과제2



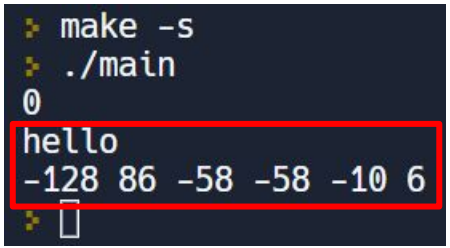
과제2

- 프로그램은 2가지 기능이 있습니다
  - 암호, 복호
- 가장 첫 입력은 option을 입력 받습니다
  - 0 : Encoding
  - 1 : Decoding



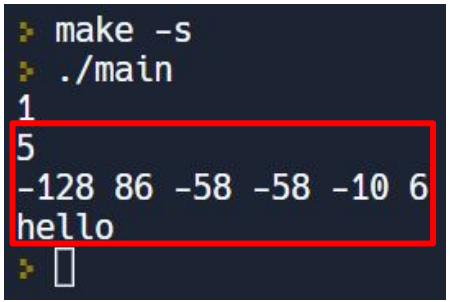
과제2

- Option 0을 선택한 경우
  - 다음의 추가적인 입력을 받습니다
    - [암호화할 문자열]
  - 이에 따른 출력은 다음과 같습니다
    - [문자 순서대로 암호화 한 결과(띄어쓰기로 구분된 정수)] [key]



과제2

- Option 1을 선택한 경우
  - 다음의 추가적인 입력을 받습니다
    - [복호할 값들의 수]
    - [띄어쓰기로 구분된 복호할 값(정수)들] [key]
  - 이에 따른 출력은 다음과 같습니다
    - [복호된 문자열]



과제2

- 암호화, 복호화 하는 문자의 개수는 최대 99개입니다
- 잘못된 입력은 일어나지 않는다고 가정합니다
- 조건을 만족하는 프로그램을 작성해보세요!

과제2 - 평가 기준

- 총 5점
- 5개의 test case 수행
- 1개 실패 마다 1점씩 감점