4190.103A-001: Programming Practice Lab. 9

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

Systems Software & Architecture Lab.

Seoul National University

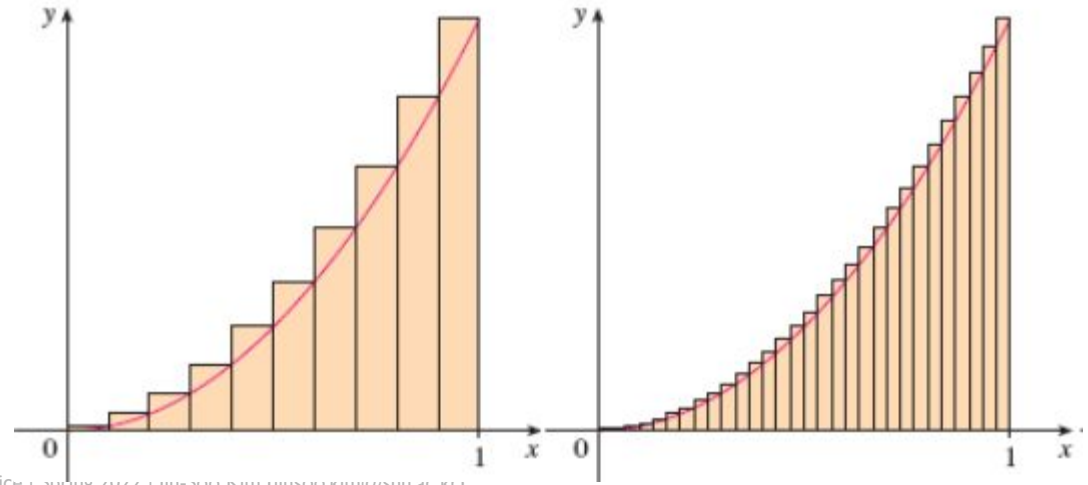
Spring 2022

과제 풀이

Lab 7 과제 1

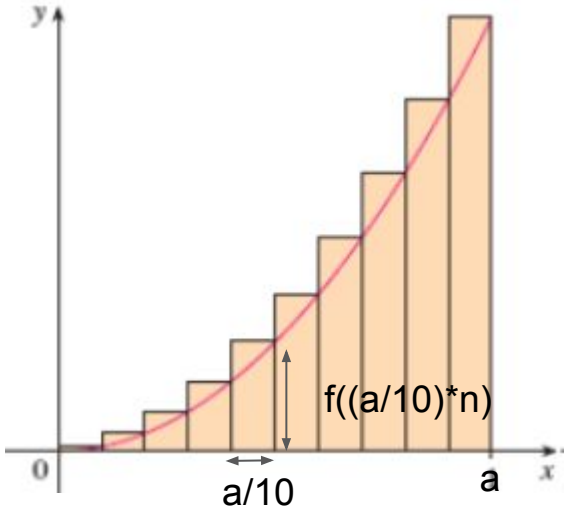
Lab 7 과제 1 – 문제

- 곡선의 넓이를 구하는 프로그램
  - 곡선의 넓이는 구분구적을 통해 구할 수 있습니다.
  - 아래 그림은 구분구적의 개념을 보여주고 있습니다.
  - 사각형의 밑변이 작아질 수록(사각형의 개수가 많아질수록) 넓이에 수렴합니다.



과제 1 – 문제

- [0,a] 사이의 f(x) = $x^2 + x$ 의 넓이를 구해봅시다
  - 입력
    - 양의 실수 a (double) (0 < a <= 100)
  - 출력
    - [0, a] 사이의 사각형의 넓이를 모두 더한 값 출력
    - 사각형의 개수 : 10, 100, 1000, 10000, 100000
    - 적분공식을 통한 계산 값
    - 모든 소수는 소수점 두번째 자리까지만 출력



$x^2 + x$의 [0,a] 적분공식
=> $(a^3/3) + (a^2/2)$

구분구적법
=> 사각형 밑변: a/10,
=> 사각형 높이: f((a/10)*n) , n은 1부터 시작
=> f(a/10)*a/10 + f(a/10*2)*a/10…. +f(a) *a/10
=> f(a/100)*a/100 + f(a/100*2)*a/100 … +f(a)*a/100

Lab 7 과제 1 – 풀이

- $f(x) = x^2 + x$

```c
#include <stdio.h>
double func(double x){
  return x*x + x;
}

double integral(double x){
  return x*x*x/3 + x*x/2;
}

double sum_areas(double x, int n){
  double length = x / n;
  double sum = 0;
  for(int i = 1; i <= n; i++)
    sum += func(length * i) * length;
  return sum;
}
```

```c
int main(void) {
  double a;
  int div[5] = {10, 100, 1000, 10000, 100000};
  scanf("%lf", &a);
  for(int i = 0; i < 5; i++) {
    printf("[%6d] %.2lf\n", div[i],
                          sum_areas(a, div[i]));
  }

  printf("%.2lf\n", integral(a));
}
```

Lab 7 과제 1 – 학생답안 1

- fun(x) = $x^2 + x$, 적분공식등 함수화 시키면 가독성이 더 좋지 않을까요?

```c
#include <stdio.h>

int main(void) {
  int square[5] = {10, 100, 1000, 10000, 100000};
  double a, temp_area=0, temp_num, temp_y, area;

  scanf("%lf", &a);
```

```c
  for (int i=0;i<5;i++){
    for (int j=0; j<square[i]; j++){
      temp_num = (a/square[i])*(j+1);
      temp_y = temp_num*temp_num +temp_num;
      temp_area += (a/square[i]) * temp_y;
    }
    printf("[%6d] %.2lf\n", square[i], temp_area);
    temp_area = 0;
  }

  area = a*a*a/3 + a*a/2;
  printf("%.2lf\n", area);
  return 0;
}
```

Lab 7 과제 1 – 학생답안 2

- 반복되는 변수 선언시 배열을 이용하고 반복되는 부분은 반복문을 사용하면 좋습니다

```c
#include <stdio.h>
double func(double a){
  return a*a + a;
}
int main(void) {
  double a;
  double sum1=0, sum2=0, sum3=0, sum4=0, sum5=0;
  scanf("%lf", &a);
  double area = a*a*a/3 + a*a/2;

  for (int i=1; i<=10; i++)
    sum1 += (a/10) * func((a/10) * i);
  for (int i=1; i<=100; i++)
    sum2 += (a/100) * func((a/100) * i);
  for (int i=1; i<=1000; i++)
    sum3 += (a/1000) * func((a/1000) * i);
  for (int i=1; i<=10000; i++)
    sum4 += (a/10000) * func((a/10000) * i);
  for (int i=1; i<=100000; i++)
    sum5 += (a/100000) * func((a/100000) * i);
  printf("[%6d] %.2f\n", 10,  sum1);
  printf("[%6d] %.2f\n", 100,  sum2);
  printf("[%6d] %.2f\n", 1000,  sum3);
  printf("[%6d] %.2f\n", 10000,  sum4);
  printf("[%6d] %.2f\n", 100000,  sum5);
  printf("%.2f\n", area);
  return 0;
}
```

Lab 7 과제 2

Lab 7 과제 2 – 문제

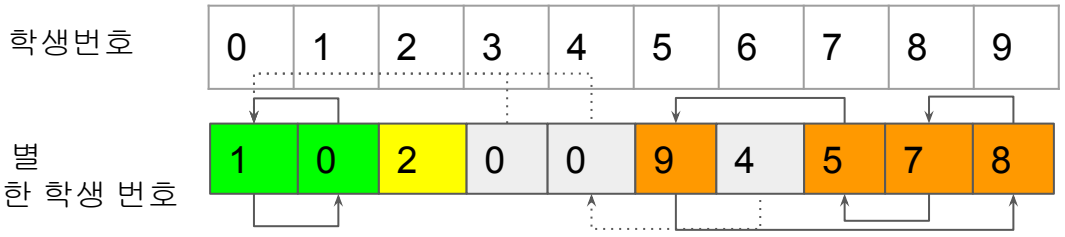
- 아래 조건을 만족하는 프로젝트팀을 구성하는 프로그램을 만들어보세요.
  - 조건
    - 학생은 10명이 있음. 각 학생의 번호는 (0 ~ 9)
    - 학생 한명당 함께하고 싶은 학생을 한명 선택함. 자기 자신을 선택해도 무방
    - 팀원 수에는 제한이 없음
    - 한 팀에 한 명만 있을 수도 있음 (자기 자신을 선택할 경우)
    - 한 팀이 되는 조건 : S1-> S2, S2 -> S3, … Sn -> S1 (최종적으로 다시 자신이 선택되어져야함)
  - 입력
    - 각 학생마다 선택한 학생의 번호를 뜻하는 정수 10개 (0 <= n <= 9)
  - 출력
    - 구성된 팀의 개수 및 각 팀의 학생 번호, 번호가 작은 학생이 포함된 팀부터, 학생번호는 작은 번호부터 출력

입력

1 0 2 0 0 9 4 5 7 8

출력

count: 3 (3팀 완성)
0 1 (0,1번 학생 같은팀)
2 (2번 학생 혼자 팀)
5 7 8 9 (5, 7, 8, 9번 학생 같은팀)



Lab 7 과제 2

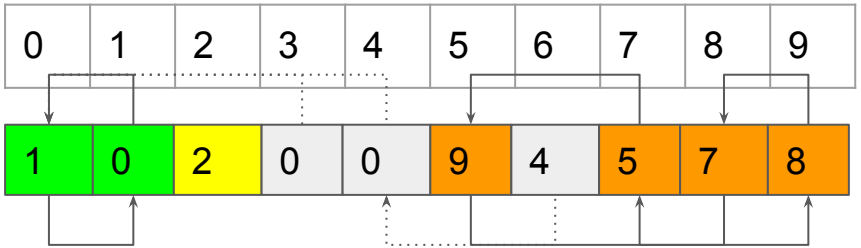
- 먼저 큰 틀을 잡아봅시다

```
int choice[ARRAY_SIZE];
int team[ARRAY_SIZE] = {0,}; //학생별 Team 번호
int team_cnt = 0; //구성된 팀 개수
for (int i = 0; i < ARRAY_SIZE; i++)
   scanf("%d", &choice[i]);

for (int student_id = 0; student_id < ARRAY_SIZE; student_id++){
   /* 팀번호가 0이면 아직 팀구성이 되지 않은 것을 의미
      팀번호가 0이 아니면 이미 팀구성이 된 학생이기 때문에 진행하지 않고 넘어감*/
   if (team[student_id] == 0){
    /* Step 1: student_id에 해당하는 학생이 팀을 이룰 수 있는지 확인 */

    /* Step 2: 팀을 이룰 수 있다면, 같은 팀으로 묶인 사람들에게 팀번호를 부여하여 팀 구성*/
   }
}
for (int team_id = 1; team_id <= team_cnt; team_id++)
   /* Step 3: 각 팀에 해당하는 학생들을 찾아서 출력함. 팀번호는 1부터 ~ team_cnt까지*/
```

Lab 7 과제 2 – Step 1 : 팀 구성 가능 확인

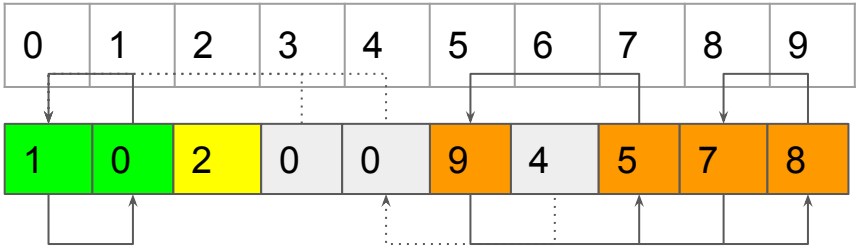
- 어떤 학생 A가 팀이 구성되려면?
  - A부터 시작해서 선택한 학생을 순회하다보면 A가 다시 나와야 합니다.

```
// 0번 학생이 팀이 구성되는지 확인하는 법
int start = 0, success = 0;
int next = choice[start];

while(1)
{
  if (start == next) {
    success = 1;
    break;
  }

  next = choice[next];
}
```



Lab 7 과제 2 – Step 1

- 학생 A가 팀이 구성되지 못하는 경우는?
  - 이미 확인한 학생을 또 다시 확인할 경우
  - 10번(총 학생수) 이상을 따라가도 자기자신이 나오지 않을 때



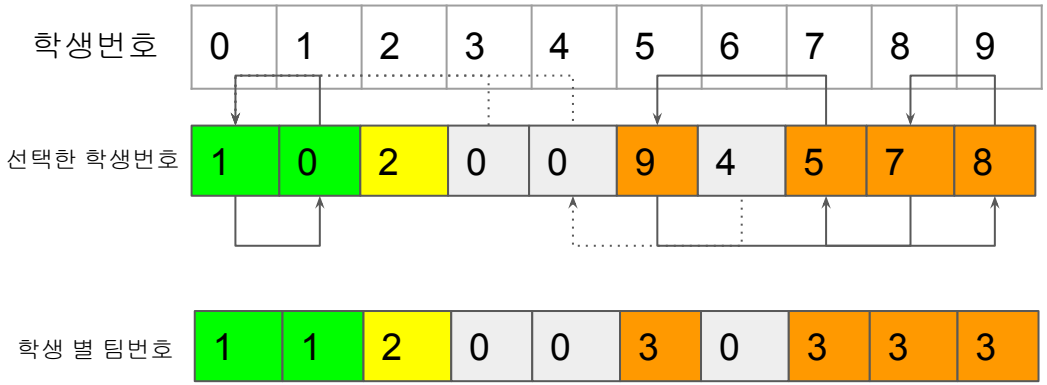
Lab 7 과제 2 – Step 1

```
// 0번 학생이 팀이 구성되는지 확인하는 법
int start = 0, success = 0;
int next = choice[start];
int visited[ARRAY_SIZE] = {0, };

while(1) {
  // 다음 학생이 이미 방문한 학생이면 0번학생은
팀구성이 안됨
  if (visited[next] == 1)
        break;
  visited[next] = 1;

  if (start == next) {
    success = 1;
    break;
  }
  next = choice[next];
}
```

Lab 7 과제 2 – Step 2 : 팀 구성

- 학생 A가 팀이 구성될 수 있다면 같은 팀으로 묶인 사람들에게 Team ID 부여하여 팀을 구성함
- Team ID의 범위 : 1 ~ team의 총개수 (0은 팀이 구성되지 않은것을 뜻함)

```
if (success == 1) {
  team_cnt++;
  for (int i = 0; i < ARRAY_SIZE; i++) {
    if (visited[i] == 1)
      team[i] = team_cnt;
}
```



Lab 7 과제 2 – Step 1, 2

- 모든 학생을 순회하며 Team이 구성되지 않은 학생만 팀을 구성할 수 있는지 확인 및 팀 구성진행

```
int choice[ARRAY_SIZE];
int team[ARRAY_SIZE] = {0,};
int team_cnt = 0;
for (int i = 0; i < ARRAY_SIZE; i++)
   scanf("%d", &choice[i]);

for (int student_id = 0; student_id < ARRAY_SIZE; student_id++){
   // team_id가 0이면 아직 팀구성이 되지 않은 것을 의미
   // team_id가 0이 아니면 이미 팀구성이 된 학생이기 때문에 진행하지 않고 넘어감
   if (team[student_id] == 0){
     // Step 1 : 팀 구성할 수 있는 지 확인
     // Step 2 : 팀 구성할 수 있다면 팀 구성 진행
   }
}
// Step 3 : 팀 개수 및 각 팀 별 학생 출력
```

Lab 7 과제 2 – Step 3 : 결과 출력

- 학생 별 팀 번호가 담겨있는 배열을 순회하면서 Team ID 1부터 마지막 번호까지 각 Team에 속해있는 학생 번호를 찾아서 출력함

```
printf("count: %d\n", team_cnt);

// team id는 1부터 ~ team_cnt까지
for (int team_id = 1; team_id <= team_cnt; team_id++) {
    for (int student_id = 0; student_id < ARRAY_SIZE; student_id++) {
      if (team[student_id] == team_id)
        printf("%d ",student_id);
    }
    printf("\n");
}
```

Lab 7 과제 1 – 학생답안

- 재귀 함수 이용

```
int findTeam(int studentNum, int mySelf, int cnt);
int studentArr[10] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
int tempArr[10] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
int student_choice[10];
int teamlist[10][10];
void sort(int arr[], int size);

int main(void) {
  int i, j, k;
  int cnt = 0, teamCount = 0;
  int hasTeam[10], memberCnt[10];

  for(i=0; i<10; i++)
    scanf("%d", &student_choice[i]);
```

```
  for(i=0; i<10; i++){
    cnt = 0;
    k = findTeam(studentArr[i], studentArr[i], cnt);
    if(k>=0){
      hasTeam[teamCount] = i;
      memberCnt[teamCount] = k;
      sort(teamlist[i], k+1);
      teamCount++;
    }
  }
  printf("count: %d\n", teamCount);

  for(i=0; i<teamCount; i++){
    for(j=0; j<=memberCnt[i]; j++){
      printf("%d ", teamlist[hasTeam[i]][j]);
    }
    printf("\n");
  }
  return 0;
}
```

Lab 7 과제 1 – 학생답안

- 재귀 함수 이용

```
int findTeam(int studentNum, int mySelf, int cnt){
  if(cnt>10 || studentNum == -1){
    return -1;
  }
  else if(student_choice[studentNum] == mySelf){
    teamlist[mySelf][cnt] = student_choice[studentNum];
    for(int i = 0; i<10; i++){
      studentArr[i] = tempArr[i];
    }
    return cnt;
  }
  else{
    teamlist[mySelf][cnt] = student_choice[studentNum];
    tempArr[student_choice[studentNum]] = -1;
   return findTeam(student_choice[studentNum], mySelf, ++cnt);
  }
}
```

Lab 8 과제 1

Lab 8 과제 1

- 괄호 문자열을 판단하는 프로그램을 만들어보세요!
- 괄호 문자열이란, ‘(‘ 와 ‘)’ 만으로 이루어진 문자열입니다
- 이 괄호 문자열은 다음의 조건을 만족합니다
  - ‘()’이 괄호 문자열의 기본 형태입니다
  - 만약, x가 괄호 문자열이라면, (x)도 괄호 문자열입니다
  - x와 y가 각각 괄호 문자열이라면, 두 괄호 문자열을 이은, xy도 괄호 문자열이 됩니다

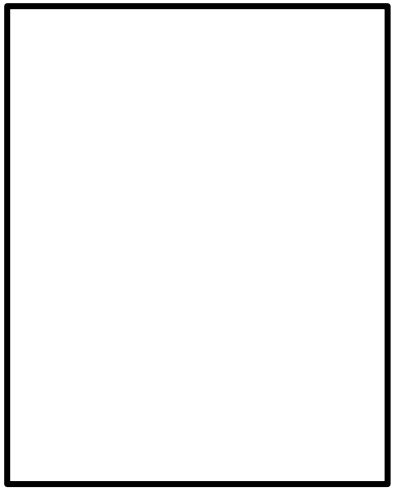
```
$ ./main
(())()
NO
```

```
$ ./main
(()())((()))
YES
```

```
$ ./main
((()()(()))(((())))()
NO
```

Lab 8 과제 1 - 설명

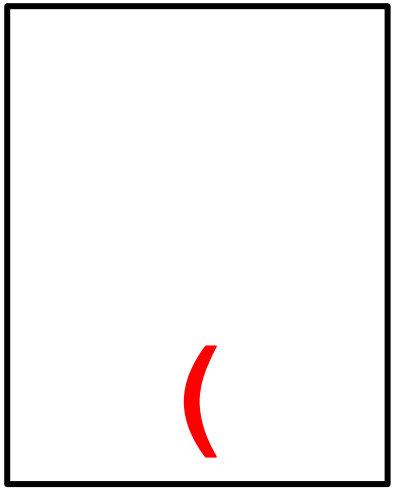
- 다음의 괄호 문자열을 판단하는 과정을 살펴봅시다!



짝을 찾는 친구들

(( ) ( ))

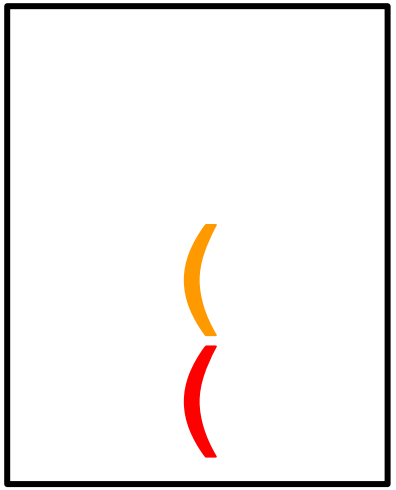
Lab 8 과제 1 - 설명



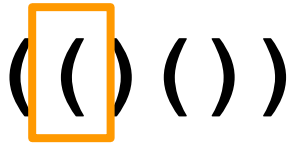
짝을 찾는 친구들

(( ) ( ))

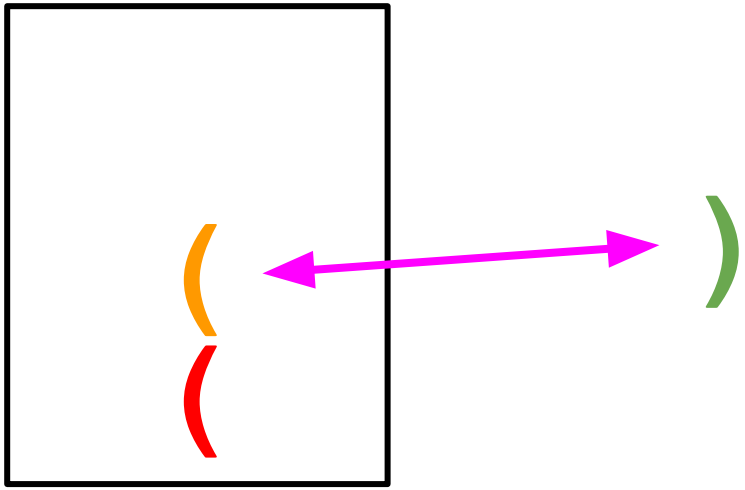
Lab 8 과제 1 - 설명



짝을 찾는 친구들

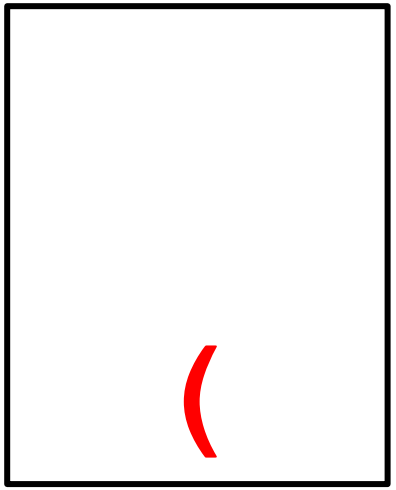


Lab 8 과제 1 - 설명



(( ) ( ))

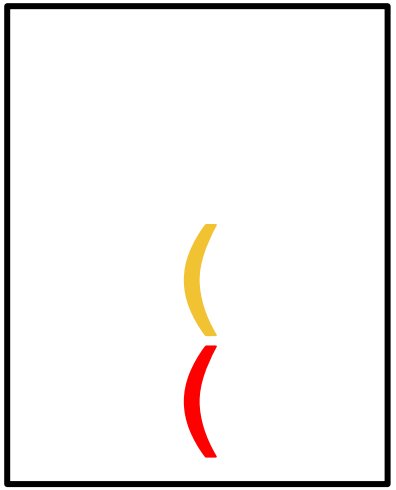
Lab 8 과제 1 - 설명



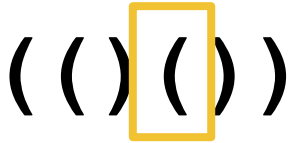
짝을 찾는 친구들

(( ) ( ))

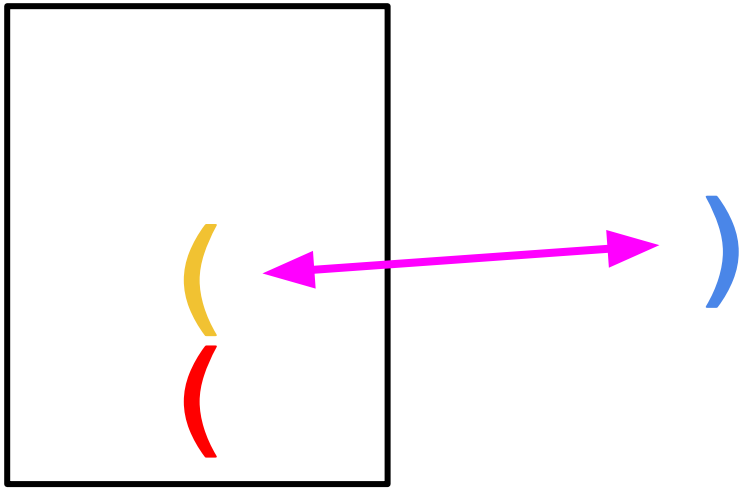
Lab 8 과제 1 - 설명

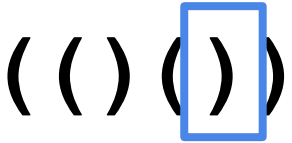


짝을 찾는 친구들

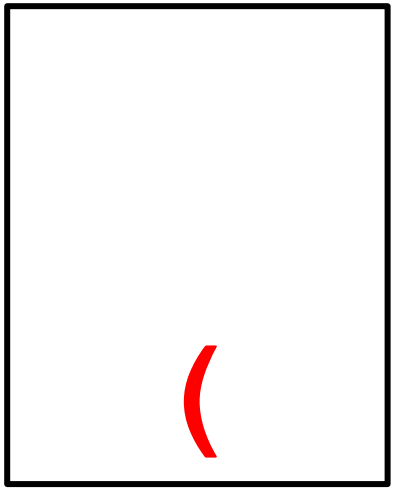


Lab 8 과제 1 - 설명





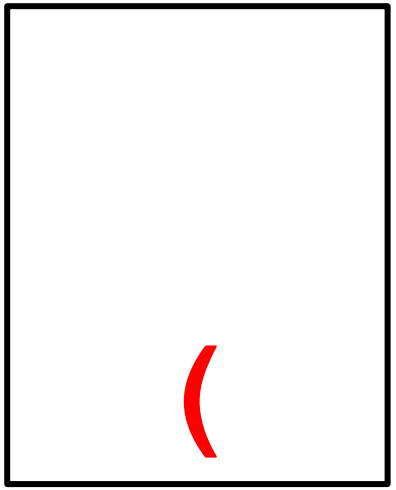
Lab 8 과제 1 - 설명



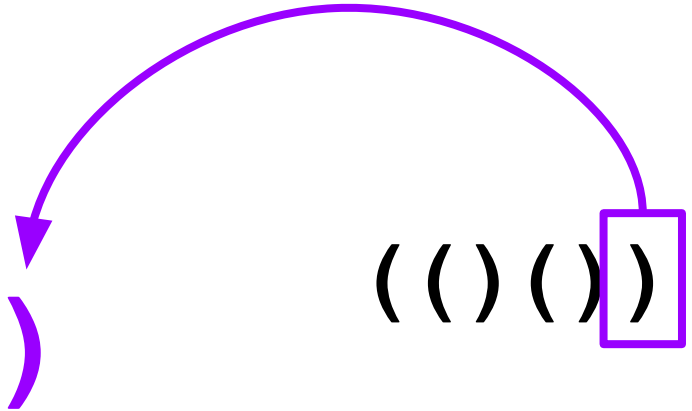
짝을 찾는 친구들

(( ) ( ))

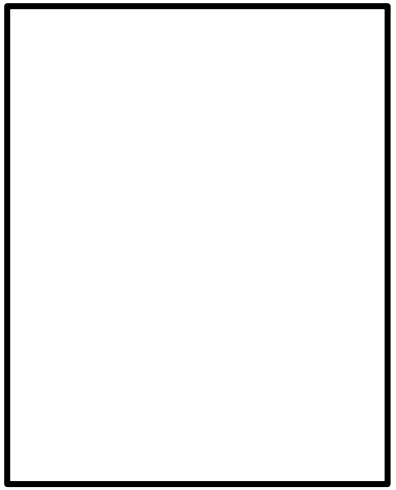
Lab 8 과제 1 - 설명



짝을 찾는 친구들



Lab 8 과제1 - 설명



짝을 찾는 친구들

(( ) ( ))

Lab 8 과제 1 – 설명

- ‘(‘ 나오면 stack에 넣고 ‘)’나오면 stack에서 하나 빼는 식으로 해서 짝이 맞는지 확인할 수 있습니다

- 본 문제는 스택과 닮아있어 스택을 이용해서 풀 수 있지만, 변수 하나만으로도 구현할 수 있습니다.
  - ‘(‘ 나오면 ++, ‘)’ 나오면 --

Lab 8 과제 1 – 풀이 1

- Stack을 이용한 풀이

```
#define SIZE 100
int stack[SIZE];
int top = -1;
int push(int i){
  if (top + 1 >= SIZE)
    return 0;
   stack[++top] = i;
  return top;
}

int pop(void){
  if (top < 0)
    return 0;
  return stack[top--];
}
```

```
int main(void) {
  int c;
  while((c = getchar()) != '\n'){
    if(c == '(')
       push('(');
    if(c == ')') {
      if (pop() != '(') {
        printf("NO\n");
        return 0;
      }
    }
  }

  if(pop() == '(')
    printf("NO\n");
  else
    printf("YES\n");

  return 0;
}
```

Lab 8 과제 1 – 풀이 2

- 변수 한개 를 이용한 방법
  - ‘(‘ 나올때 ++, ‘)‘ 나올때 --

```c
int main(void) {
  int c, open = 0;
  while((c = getchar()) != '\n'){
    if(c == '(')
        open++;
    if(c == ')') {
      if (open == 0) {
        printf("NO\n");
        return 0;
      }
      else
        open--;
    }
  }

  if(open > 0)
    printf("NO\n");
  else
    printf("YES\n");

  return 0;
}
```

Lab 8 과제 1 – 학생 답안

- 변수를 이용한 풀이법
- ‘(‘ 의 개수와 ‘)’개수를 구분하여 횟수를 셈

```
int main() {
  int c, a = 0, b = 0;
  while ((c = getchar()) != '\n'){
    if (c == 40) // putchar(40) = '('
      a++;
    else
      b++;

    if (a == 0 && b == 1){
      break;
    }
    else if (b == 1){
        a--;
        b--;
    }
  }

  if (a == 0 && b == 0)
    printf("YES\n");
  else
    printf("NO\n");
}
```

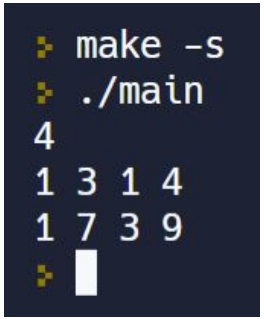
Lab 8 과제 2

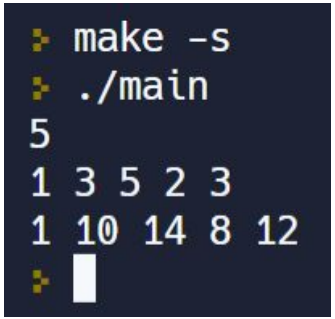
Lab 8 과제 2 – 설명

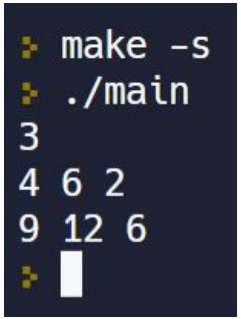
- 다음 문제를 해결할 수 있는 프로그램을 만들어보세요!
- 학생들에게 피자를 나누어주려고 합니다
- 학생들은 줄을 서서 순서대로 피자를 한 조각씩 받습니다
- 피자를 더 먹고 싶은 학생은 줄의 마지막으로 가서 다시 설 수 있습니다
- 각 학생들이 자신이 먹고 싶은 만큼 피자를 받으려면 얼만큼의 시간이 걸리는지 구해보세요!

Lab 8 과제 2 – 설명

- 입력은 다음과 같습니다
  - 첫 번째 입력은 피자를 받을 학생의 수를 나타냅니다
  - 두 번째 입력은 피자를 받는 순서대로, 각 학생이 먹고 싶은 피자 조각의 수를 나타냅니다
- 결과는 각 학생들이 자신이 원하는 만큼 받기 위해서 얼만큼 기다려야 하는지 순서대로 출력하면 됩니다







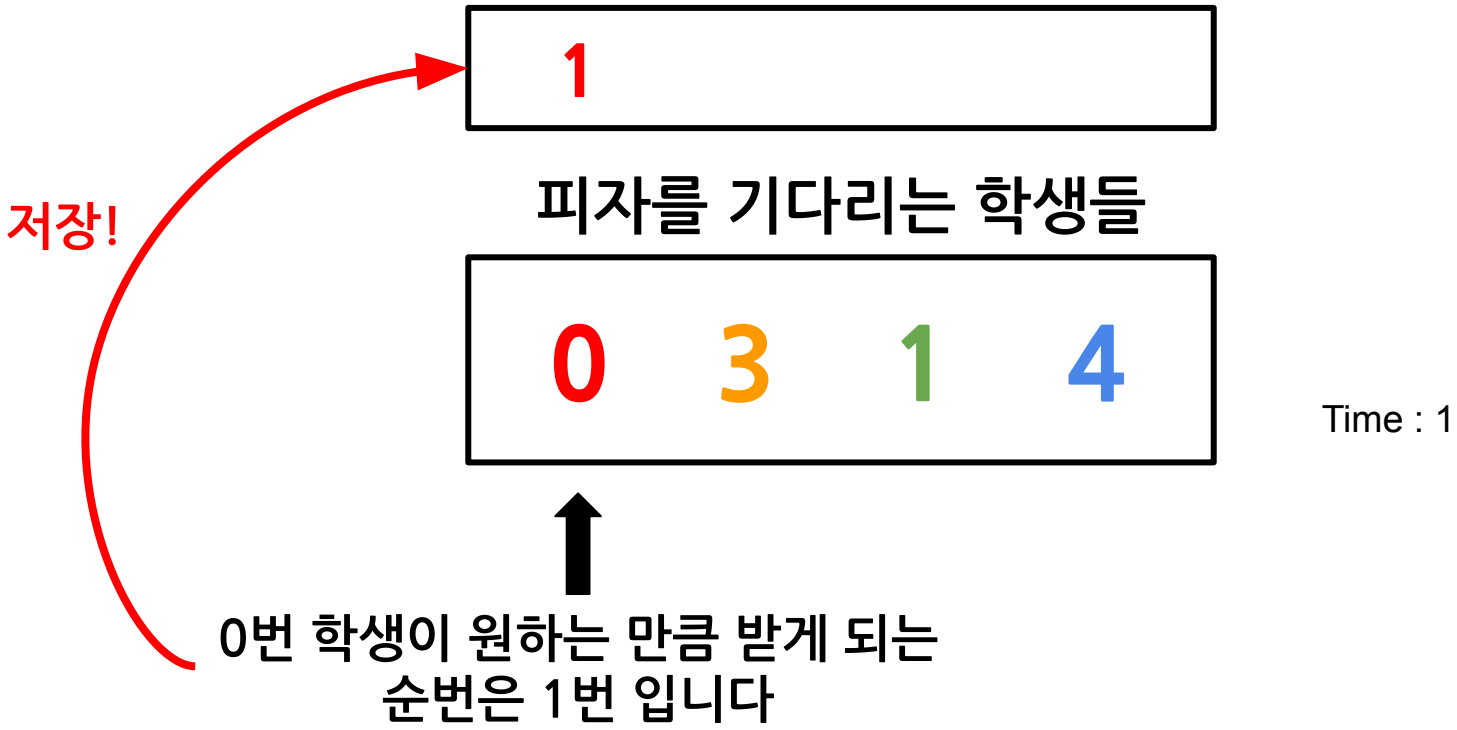
Lab 8 과제 2 – 설명

- 마찬가지로 문제 해결을 위한 아이디어를 얻기 위해 문제를 살펴봅시다!

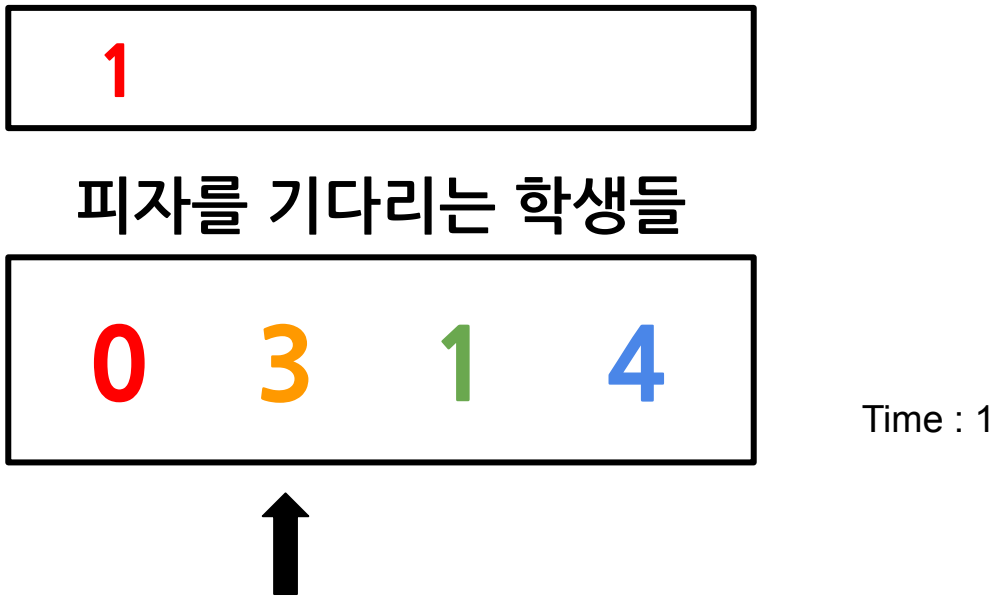
피자를 기다리는 학생들

Lab 8 과제 2 – 설명

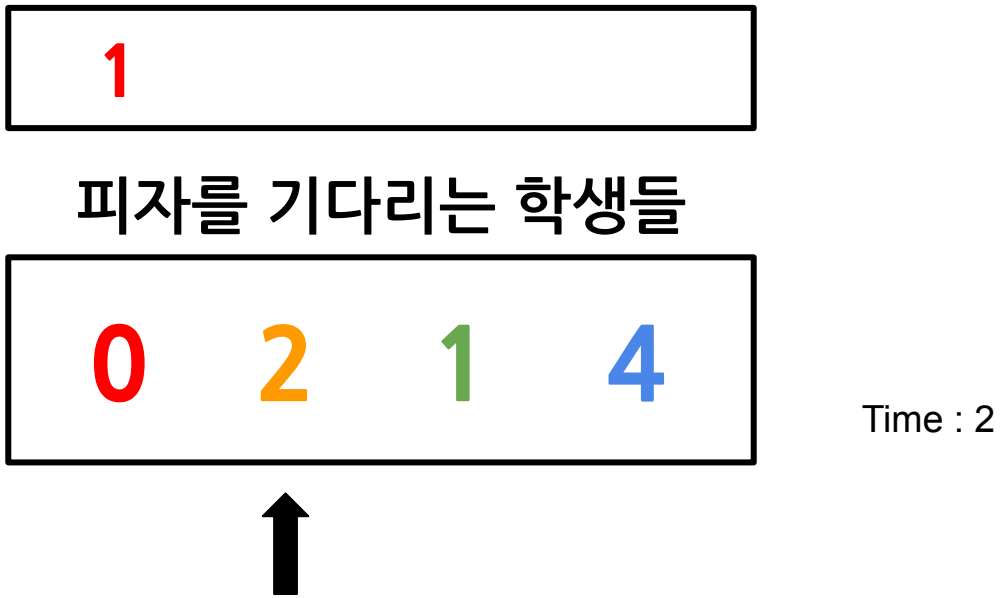
Lab 8 과제 2 – 설명



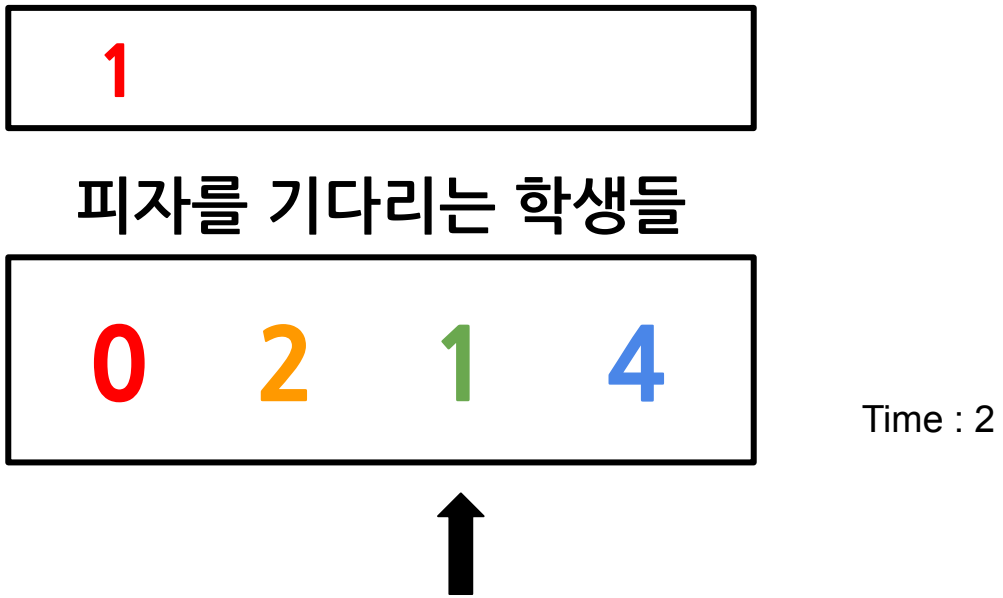
Lab 8 과제 2 – 설명



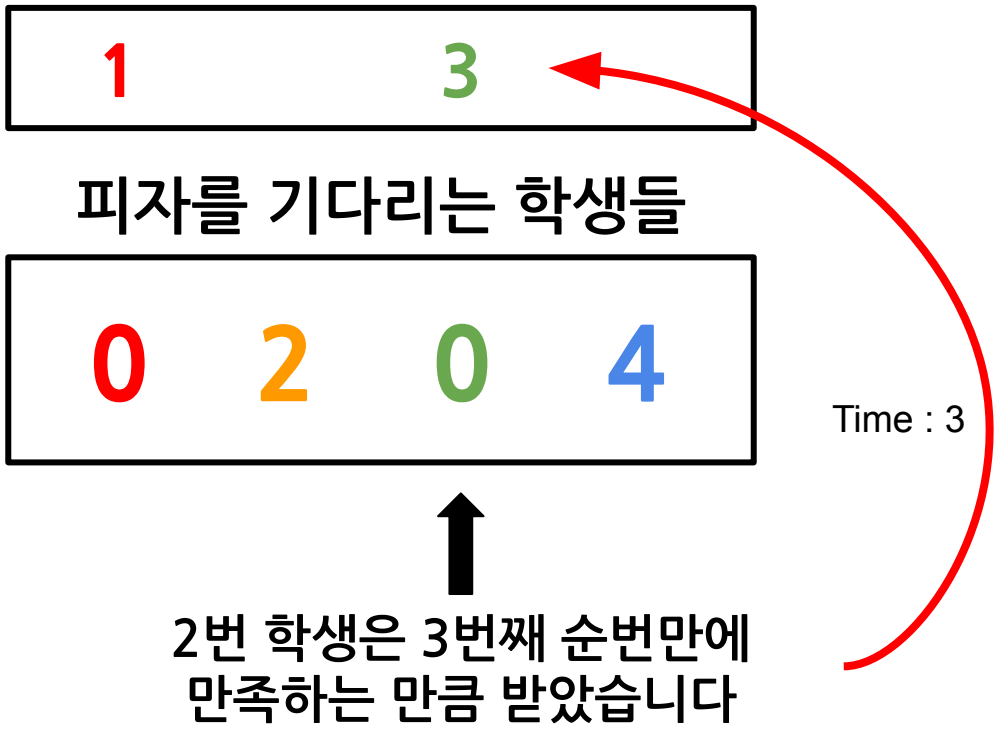
Lab 8 과제 2 – 설명



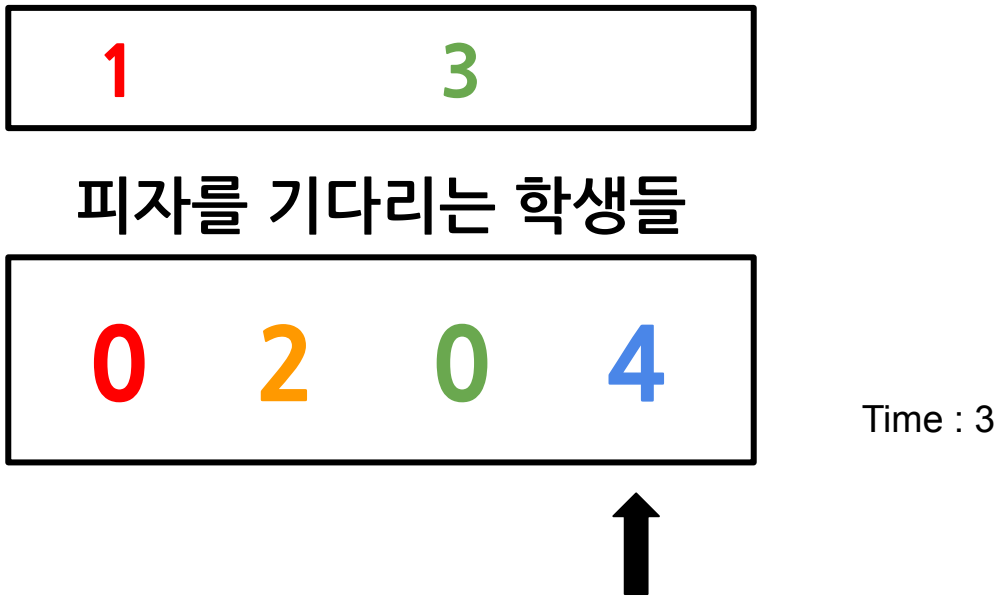
Lab 8 과제 2 – 설명



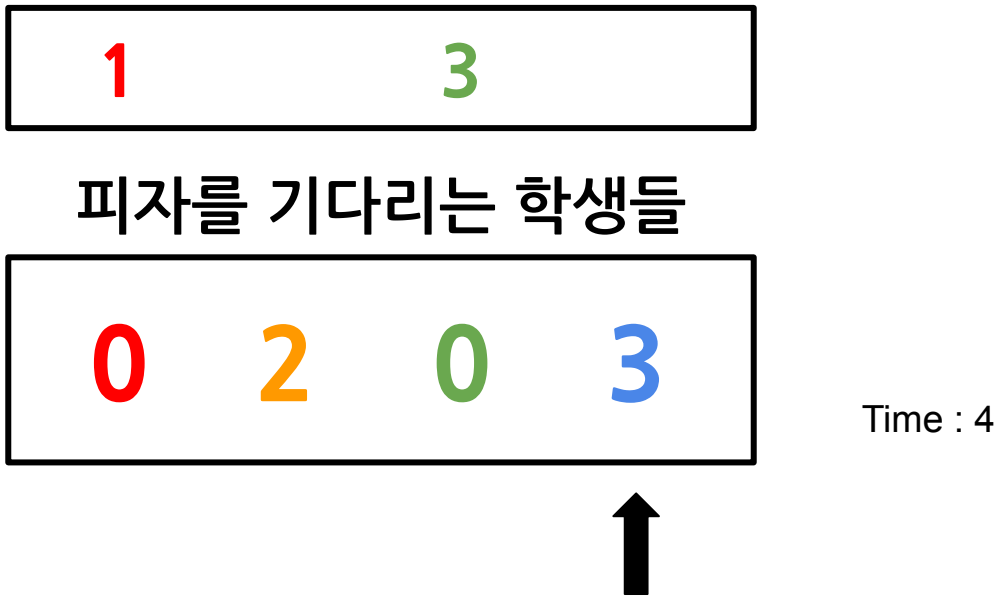
Lab 8 과제 2 – 설명



Lab 8 과제 2 – 설명

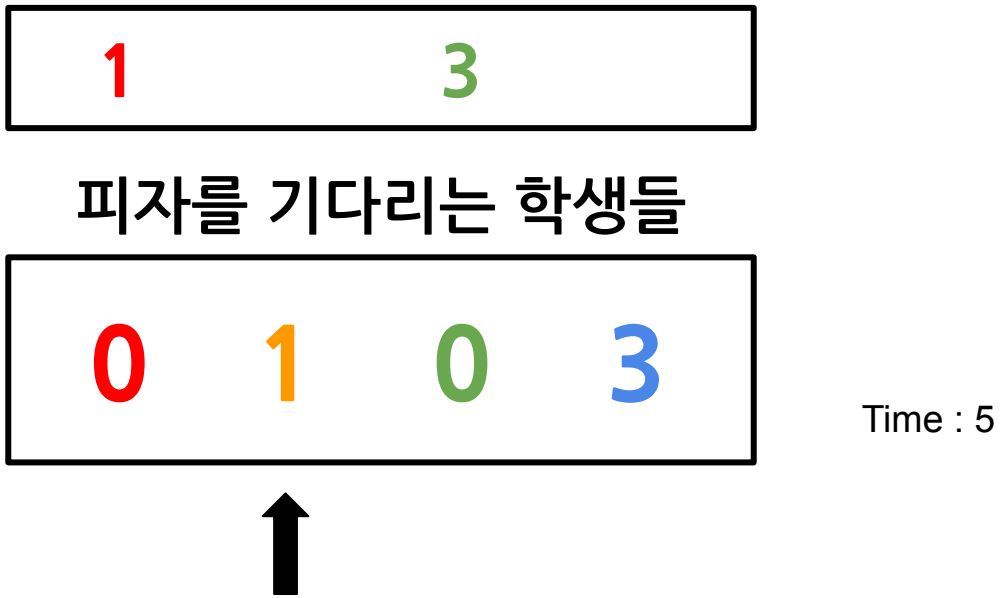


Lab 8 과제 2 – 설명



Lab 8 과제 2 – 설명

Lab 8 과제 2 – 설명



Lab 8 과제 2 – 풀이

```c
#include <stdio.h>
#define MAX 10

int main(void) {
  int remaining_pizza[MAX];
  int time_taken[MAX] = {0, };

  int stu_id = 0, stu_num, time = 0;
  int remaining_people = 0;

  scanf("%d", &stu_num);
  for(int i = 0; i < stu_num; i++)
    scanf("%d", &remaining_pizza[i]);

  remaining_people = stu_num;
  while(remaining_people > 0){
    if (remaining_pizza[stu_id] > 0) {
      time++;
      remaining_pizza[stu_id]--;

      if(remaining_pizza[stu_id] == 0) {
        time_taken[stu_id] = time;
        remaining_people--;
      }
    }
    stu_id = (stu_id + 1) % stu_num;
  }

  for(int i = 0; i < stu_num; i++)
    printf("%d ", time_taken[i]);

  return 0;
}
```

Lab 8 과제 2 – 학생 답안 1

- 피자를 먹고자 남아있는 학생을 Queue를 통해 표현함
- 초기에 Queue에 모든 학생을 순차적으로 넣음
- Queue에서 한명 빼서 피자를 먹이고, 이 학생이 더 먹고자 한다면 다시 Queue에 넣어주는 방식

```
int main(void) {
  int n, t = 0; want[10], time[10];
  scanf("%d", &n);
  for (int i = 0; i < n; i++)
    scanf(" %d", &want[i]);

  for (int i = 0; i < n; i++)
    enqueue(i);

  for (t = 1; !isEmpty(); t++) {
    int curr = dequeue();
    want[curr]--;
    if (want[curr] > 0) {
      enqueue(curr);
    } else {
      time[curr] = t;
    }
  }

  for (int i = 0; i < n; i++)
    printf("%d ", time[i]);
}
```

Lab 8 과제 2 – 학생 답안 2

- 들여쓰기를 일정하게 사용하고, 변수의 의도 및 이름을 명확히 해야지 가독성이 좋아집니다

```
while(1){
    for(int i = 0;  i < student_num ; i++){
      for(int j = 0; j < student_num; j++){
        if(pizza[j] < queue[j])
          count[j]++;
        }
      if(pizza[i] < queue[i])
        pizza[i]++;
      }
     for(int i = 0;  i < student_num ;i++){
      if(pizza[i] < queue[i])
        count[i] -= front;
      }
```

```
    for(int i = 0; i < student_num ; i++){
      if(arr[i] == 0){
        if(pizza[i] == queue[i]){
          arr[i]++;
          front++;

          for(int j = i + 1; j < student_num; j++)
            if(pizza[j] < queue[j])
              arr2[j]++;
        }
      }
    }
    if(front >= student_num)
      break;
    }

  for(int i = 0; i < student_num ; i++)
     count[i] -= arr2[i];
```

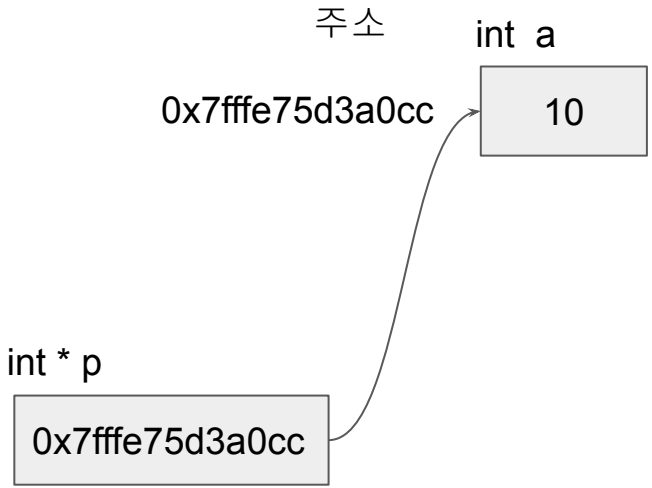
실습 1

실습 1 – 설명

- 포인터
  - & : 주소연산자
  - * : 참조연산자

```
int a;
int *p;
p = &a; // a변수의 주소값을 p에 저장
*p = 10; // p가 가리키는 위치의 데이터를 10으로 씀

printf("a: %d, *p: %d, p: %p\n", a, *p, p);
```

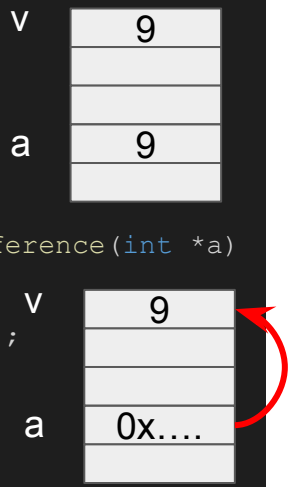
```
> ./main
a: 10, *p: 10, p: 0x7ffe75d3a0cc
```



실습 1 – 설명

- Call by value vs Call by reference

```
void square_by_value(int a)
{
  a = a * a;
  return;
}

void square_by_reference(int *a)
{
  *a = (*a) * (*a);
  return;
}
```




```
int main(void)
{
  int v = 9;

  square_by_value(v);
  printf("[call by value] v: %d\n", v);

  square_by_reference(&v);
  printf("[call by reference] v: %d\n",v);
}
```

```
./main
[call by value] v: 9
[call by reference] v: 81
```

실습 1

- 인자로 받은 포인터에 결과를 넘기는 함수를 만들어보세요
  - void sum(int a, int b, int *result);
  - void sub(int a, int b, int *result);
  - void swap_and_sub(int *a, int *b, int *result);
    - 결과값 : 큰 값 – 작은 값
    - b가 가리키는 값이 a보다 크다면 swap, 아니면 유지

```
> ./main
1 10
[sum] a: 1 b: 10 result: 11
[sub] a: 1 b: 10 result: -9
[swap_and_sub] a: 10 b: 1 result: 9
>
> ./main
10 1
[sum] a: 10 b: 1 result: 11
[sub] a: 10 b: 1 result: 9
[swap_and_sub] a: 10 b: 1 result: 9
```

실습 1 – 풀이

```c
void sum(int a, int b, int *result){
  *result = a + b;
}

void sub(int a, int b, int *result){
  *result = a - b;
}

void swap_and_sub(int *a, int *b, int
*result) {
  if (*a < *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
  }

  *result = *a - *b;
}
```

```c
int main(void)
{
  int a,b;
  int result;
  scanf("%d %d", &a, &b);

  sum(a, b, &result);
  printf("[sum] a: %d b: %d result: %d\n", a, b, result);

  sub(a, b, &result);
  printf("[sub] a: %d b: %d result: %d\n", a, b, result);

  swap_and_sub(&a, &b, &result);
  printf("[swap_and_sub] a: %d b: %d result: %d\n", a, b, result);
}
```

실습 1 – 설명

- Array and Pointer

```c
#define SIZE 5
int main(void) {
  int arr[SIZE] = {0,};
  int *p = arr; // 배열이름 -> 배열의 시작주소
  printf("p: %p arr: %p\n", p, arr);

  for (int i = 0; i < SIZE; i++) {
    //  arr + i <->  &arr[i]
    printf("p+%d: %p &arr[%d]: %p\n",i, p+i, i, &arr[i]);
    *(p+i) = i; // arr[i] = i;
  }
  for (int i = 0; i < SIZE; i++)
    // *(arr + i) <->  arr[i]
    printf("*(p+%d): %d arr[%d]: %d\n",i, *(p+i), i, arr[i]);
}
```

		주소	arr[5]
&arr[0]	arr + 0	0x7ffea4aaf2b0	0
&arr[1]	arr + 1	0x7ffea4aaf2b4	1
	arr + 2	0x7ffea4aaf2b8	2
	arr + 3	0x7ffea4aaf2bc	3
	arr + 4	0x7ffea4aaf2c0	4

```
./main
p: 0x7ffea4aaf2b0 arr: 0x7ffea4aaf2b0
p+0: 0x7ffea4aaf2b0 &arr[0]: 0x7ffea4aaf2b0
p+1: 0x7ffea4aaf2b4 &arr[1]: 0x7ffea4aaf2b4
p+2: 0x7ffea4aaf2b8 &arr[2]: 0x7ffea4aaf2b8
p+3: 0x7ffea4aaf2bc &arr[3]: 0x7ffea4aaf2bc
p+4: 0x7ffea4aaf2c0 &arr[4]: 0x7ffea4aaf2c0
*(p+0): 0 arr[0]: 0
*(p+1): 1 arr[1]: 1
*(p+2): 2 arr[2]: 2
*(p+3): 3 arr[3]: 3
*(p+4): 4 arr[4]: 4
```

실습 1 – 제출

- concat(), print_array()함수 및 빈칸을 완성해보세요
  - a[] 뒤에 b[]을 연결하는 함수. 결과는 c[]에 넣음
  - n은 a[]의 size, m은 b[]의 size

```
> ./main
1 2 3 4 5 6 7 8 9 10
```

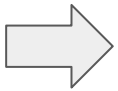
```c
#include <stdio.h>
#define MAX_ARRAY_SIZE 100

void print_array(int a[], int n) {
  /*채워보세요*/
}

void concat(int a[], int n, int b[], int m, int c[]){
  /*채워보세요*/
}
```

```c
int main(void) {
  int arr1[] = {1, 2, 3, 4, 5};
  int arr2[] = {6, 7, 8, 9 ,10};
  int arr3[MAX_ARRAY_SIZE] = {0, };
  int size1 = sizeof(arr1)/sizeof(int);
  int size2 = sizeof(arr2)/sizeof(int);
  /*arr1 뒤에 arr2를 붙여 arr3에 저장하세요*/
  concat(/* 빈칸 */);

  /*arr3를 출력하세요*/
  print_array(/* 빈칸 */);
  return 0;
}
```

실습 1 – 제출

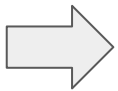
- 아래 함수에서 [ ]연산자를 * 연산자로 바꿔보세요

```
void print_array(int a[], int n) {
  for(int i = 0; i < n; i++)
    printf("%d ", a[i]);
  printf("\n");
}
```



```
void print_array(int * a, int n){
  /*포인터 연산을 사용해보세요*/
}
```

```
void concat(int a[], int n, int b[], int m, int c[]){
  for (int i = 0; i < n+m; i++){
    if (i < n)
      c[i] = a[i];
    else
      c[i] = b[i-n];
  }
}
```



```
void concat(int * a, int n, ....){
  /*포인터 연산을 사용해보세요*/
}
```

실습 1 – 풀이

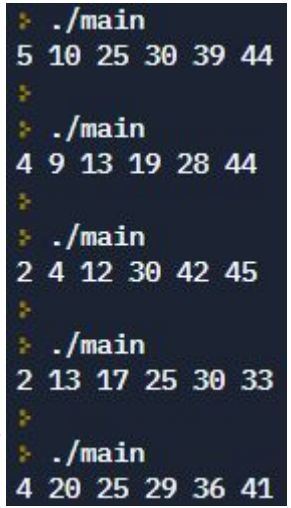
```c
void print_array(int *a, int n){
  for(int i = 0; i < n; i++)
    printf("%d ", *(a+i));

  printf("\n");
}

void concat(int *a, int n, int *b, int m, int *c){
  for (int i = 0; i < n+m; i++){
    if (i < n)
      *(c+i) = *(a+i);
    else
      *(c+i) = *(b+i-n);
  }
}
```

```c
int main(void) {
  int arr1[] = {1, 2, 3, 4, 5};
  int arr2[] = {6, 7, 8, 9 ,10};
  int arr3[MAX_ARRAY_SIZE] = {0, };

  int size1 = sizeof(arr1)/sizeof(int);
  int size2 = sizeof(arr2)/sizeof(int);

  concat(arr1, size1, arr2, size2, arr3);

  print_array(arr3, size1 + size2);
  return 0;
}
```

실습 2

실습 2 – 제출

- 1부터 45까지 6개의 임의의 로또번호를 생성하는 프로그램을 작성하세요
  - 중복된 값이 있으면 안됨
  - 실행할때마다 새로운 번호를 생성해야 함
  - 오름차순으로 정렬해야함
  - Random 함수 이용
    - srand(time(NULL));
    - rand();
  - 포인터 연습을 위해 []연산 대신 포인터관련 연산(*, &)을 사용해주세요



실습 2 – Step 1, 2

- Step 1, 2를 채워보세요

```c
#define NUM_LOTTO 6

/* 기존 나온 번호인지 확인하는 함수. */
int check_duplicate_number(int * lotto, int count,
                                        int number){}

/* 번호를 추가하는 함수. 오름차순으로 중간에 삽입함. count증가*/
void insert_number(int * lotto, int * count,
                                int  number){}

void print_lotto(int * lotto, int count) {
  for (int i = 0; i < count; i++)
    printf("%d ", *(lotto + i));
  printf("\n");
}
```

```c
int main(void) {
  int lotto[NUM_LOTTO];
  int count = 0, selected_number;
  srand(time(NULL));
  do {
    /* Step #1: 1 ~ 45 사이의 임의의 번호 선택 */

    /* Step #2: 기존 나온 번호인지 확인 후 그렇다면
                다시 Step #1으로 돌아감  */

    /* Step #3: 선택된 번호를 lotto[]배열에
                오름차순으로  삽입 */
  } while (/* 빈칸 */);
  print_lotto(/* 빈칸 */);
  return 0;
}
```

실습 2 – Step 1, 2 – 풀이

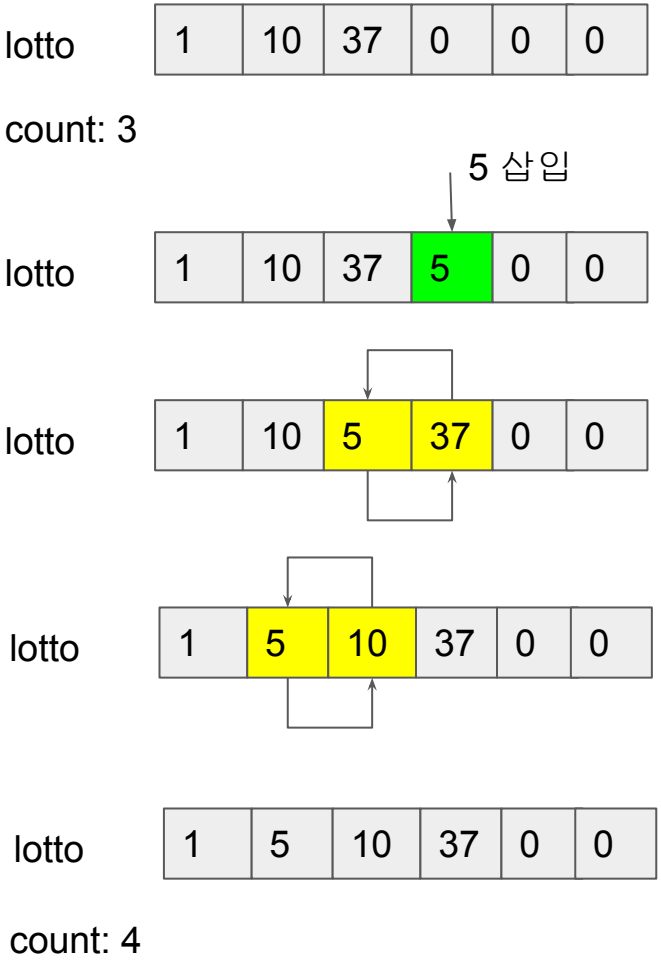
```c
/* 기존 나온 번호인지 확인하는 함수. */
int check_duplicate_number(int *lotto, int count, int
number){
  for (int i = 0; i < count; i++) {
    if (*(lotto+i) == number)
      return 1;
  }
   return 0;
}
```

```c
int main(void) {
  int lotto[NUM_LOTTO];
  int count = 0, selected_number;
  srand(time(NULL));

  do {
    selected_number = rand()%45 + 1;
    if (check_duplicate_number(lotto, count,
                               selected_number))
      continue;

    insert_number(lotto, &count,selected_number);
  } while (count < NUM_LOTTO);

  print_lotto(lotto, count);
  return 0;
}
```

실습 2 – Step 3

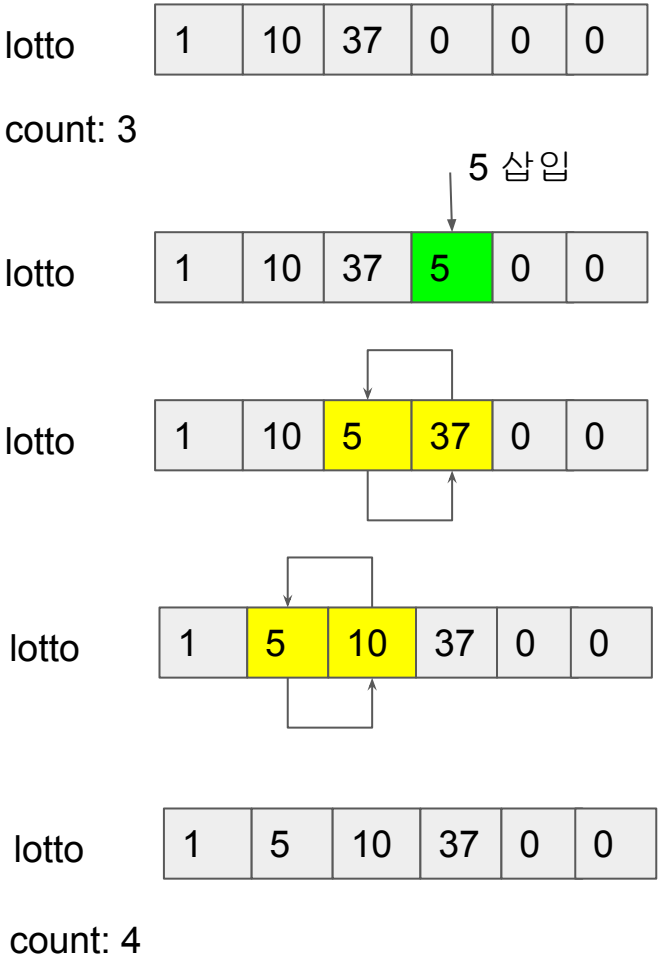
- Insert_number() 함수를 작성해보세요
  - 오름차순으로 어떻게 중간에 넣을 수 있을까요?

```c
/* 번호를 추가하는 함수 */
void insert_number(int *lotto, int *count, int number)
{
  /* 채워보세요 */
  /* 오름차순으로 중간에 삽입함 */
  /* count증가 */
}
```



실습 2 – Step 3 – 풀이

```c
void swap(int * a, int * b) {
  int temp = *a;
  *a = *b;
  *b = temp;
}

void insert_number(int *lotto,int *count,int
number){
  int i, cnt = *count;
  *(lotto + cnt) = number;

  for (i = cnt; i > 0; i--) {
    if (*(lotto + i) < *(lotto + i - 1))
      swap(lotto + i, lotto + i - 1);
    else
      break;
  }
  cnt++;
  *count = cnt;
}
```

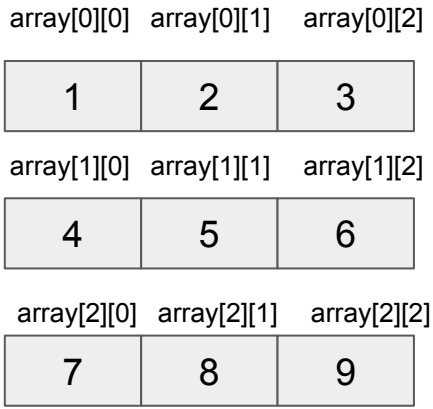


실습 3

실습 3 – 설명

- 다차원배열

```c
#include <stdio.h>
int main(void) {
  // 3X3 배열 선언 및 초기화
  int array[3][3] = {{1,2,3}, {4,5,6}, {7,8,9}};

  // 3X3 배열 순회 및 element 출력
  for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 3; j++)
      printf("%d ", array[i][j]);
    printf("\n");
  }
  /* 반복문을 이용하여 모든 element의 주소도 출력해보세요 */
  /* printf("%p\n", &array[i][j]);*/
  return 0;
}
```



실습 3

- 3층, 4라인 아파트의 사람수를 파악하는 프로그램을 만들어봅시다.
  - 각 층 별로 각 라인의 사람수를 입력 받음 (총 12개)
  - 각 라인의 총 사람수, 각 층의 총 사람수를 출력
  - 층 수와 라인수는 #define으로 선언
    - #define NUM_FLOORS 3
    - #define NUM_LINES 4

	1라인	2라인	3라인	4라인	층별 총합
1층	1	1	1	1	4명
2층	2	4	1	1	8명
3층	3	0	5	4	12명
라인 총합	6명	5명	7명	6명	

```
> ./main
1 1 1 1
2 4 1 1
3 0 5 4
[1 floor] # people: 4
[2 floor] # people: 8
[3 floor] # people: 12
[1 line] # people: 6
[2 line] # people: 5
[3 line] # people: 7
[4 line] # people: 6
```

실습 3

- 함수를 만들어 사용해주세요

```
int count_people_on_floor(int array[][NUM_LINES], int floor)
{

}

int count_people_on_line(int array[][NUM_LINES], int line)
{

}
```

실습 3 – 풀이

```c
#define NUM_FLOORS 3
#define NUM_LINES 4

int count_people_on_floor(int array[][NUM_LINES], int floor)
{
  int num_people = 0;
  for (int i =0; i < NUM_LINES; i++)
    num_people += array[floor][i];

  return num_people;
}

int count_people_on_line(int array[][NUM_LINES], int line)
{
  int num_people = 0;
  for (int i =0; i < NUM_FLOORS; i++)
    num_people += array[i][line];

  return num_people;
}
```

```c
int main(void) {
  int array[NUM_FLOORS][NUM_LINES];

  for (int floor = 0; floor < NUM_FLOORS; floor++)
    for (int line =0; line < NUM_LINES; line++)
      scanf("%d", &array[floor][line]);

  /* 각 층별 사람 총합 출력 */
  for (int floor = 0; floor < NUM_FLOORS; floor++)
    printf("[%d floor] # people: %d\n", floor+1,
                    count_people_on_floor(array, floor));

  /* 각 라인별 사람 총합 출력 */
  for (int line = 0; line < NUM_LINES; line++)
    printf("[%d line] # people: %d\n", line+1,
                    count_people_on_line(array, line));

  return 0;
}
```

실습 3 – Tip

- NUM_FLOORS와 NUM_LINES를 변경하여 실행해보세요
  - #define을 잘 활용하면 아래의 상황과 같이 프로그램의 요구사항이 바뀔 때 작은 수정을 통해 쉽게 프로그램을 변경할 수 있습니다.

```
#define NUM_FLOORS 2
#define NUM_LINES 2
```

```
> ./main
1 3
2 4
[1 floor] # people: 4
[2 floor] # people: 6
[1 line] # people: 3
[2 line] # people: 7
```

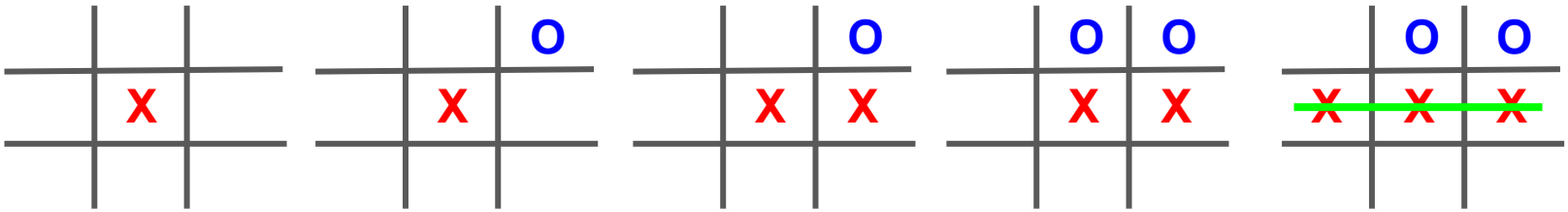
```
#define NUM_FLOORS 5
#define NUM_LINES 2
```

```
> ./main
1 1
2 2
3 3
4 4
5 5
[1 floor] # people: 2
[2 floor] # people: 4
[3 floor] # people: 6
[4 floor] # people: 8
[5 floor] # people: 10
[1 line] # people: 15
[2 line] # people: 15
```

실습 4

실습 4

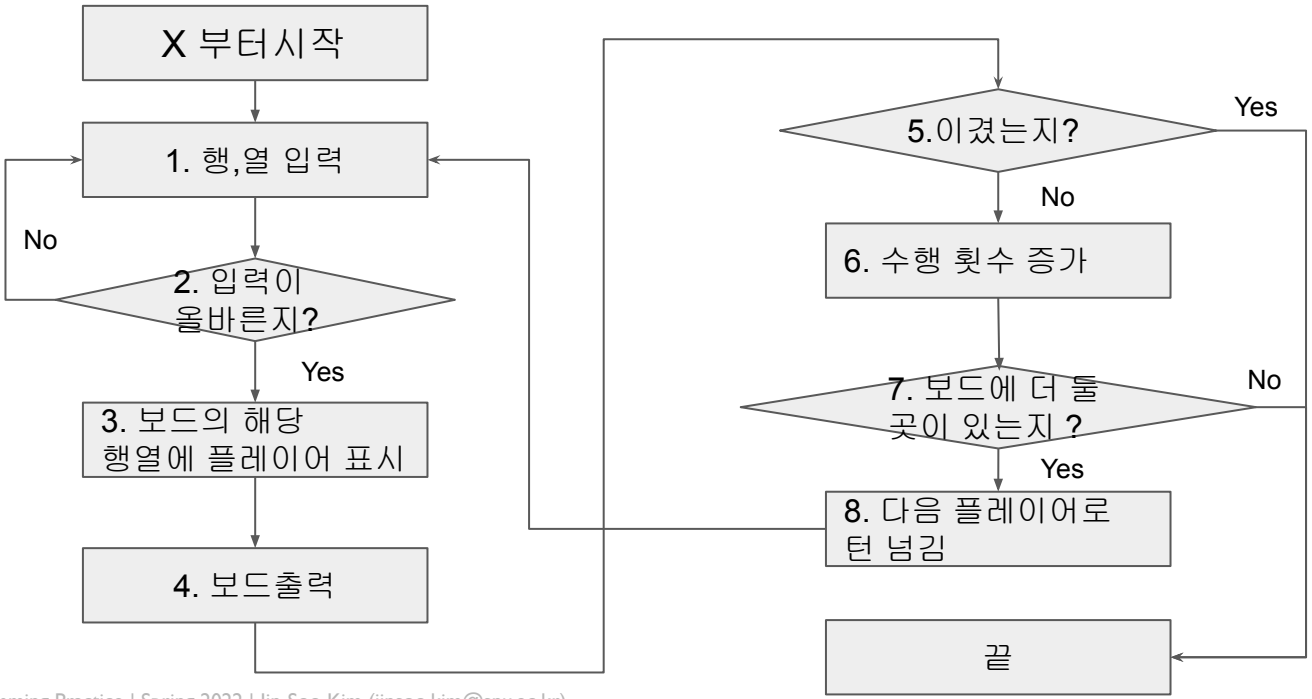
- Tic-Tac-Toe Game
  - 3x3 board에 두명이 번갈아가며 X와 O를 써서 같은 글자를 가로, 세로 혹은 대각선 상에 놓이면 이기는 게임
  - 무승부가 될 수도 있음



X가 이김

실습 4

- 코딩하기 전에 프로그램의 큰 틀을 먼저 잡아봅시다



실습 4 – 큰 틀 잡기

- 무작정 코딩부터 진행하기보다 알고리즘을 먼저 생각해보고 큰 틀을 만들어보면 복잡한 프로그램도 쉽게 구현할 수 있습니다
- 함께 하나하나 차근차근 만들어가 봅시다

@main()

```
  /* 1. 보드 전체 출력  */
  while (1) {
    printf("Turn[%c] (row column): ", turn);
    scanf("%d %d", &row, &column);
    printf("\n");

    /* 2. 입력이 올바른지 확인
          올바르지 않다면 continue 사용하여 다시 입력받음*/

    /* 3. 보드의 입력받은 위치에 플레이어 표시
         X는 1, O는 -1로 표시함*/

    /* 4. 보드 전체 출력 */

    /* 5 : 이겼는지 확인
           이겼으면 이긴 Player 출력하고 반복문 빠져나옴*/

    /* 6 : 총 횟수 증가 및 더 이상 둘 곳이 있는지 확인.
           더이상 둘 곳이 없다면 반복문 빠져나옴*/

    /* 7 : 플레이어 변경. X->O, O->X*/
  }
}
```

실습 4 – Step 1

- 먼저 단순하게 반복문을 돌면서 입출력해주는 부분부터 시작하겠습니다.
- #3, #6, #7을 완성해주세요(@main)

```c
#include <stdio.h>
#define LENGTH 3

int main(void) {
  int board[LENGTH][LENGTH] = { 0,};
  char turn = 'X';
  int count = 0, row, column;
  /* #1. 보드 전체 출력  */
  while (1) {
    printf("Turn[%c] (row column): ", turn);
    scanf("%d %d", &row, &column);
    printf("\n");

    /* #2. 입력이 올바른지 확인
          올바르지 않다면 continue 사용하여 다시 입력받음*/

    /* #3. 보드의 입력받은 위치에 플레이어 표시
         X는 1, O는 -1로 표시함*/

    /* #4. 보드 전체 출력 */

    /* #5 : 이겼는지 확인
          이겼으면 이긴 Player 출력하고 반복문 빠져나옴*/

    /* #6 : 총 횟수 증가 및 더 이상 둘 곳이 있는지 확인.
          더이상 둘 곳이 없다면 반복문 빠져나옴*/

    /* #7 : 플레이어 변경. X->O, O->X*/
  }
}
```

실습 4 – Step 1

```c
#include <stdio.h>
#define LENGTH 3

int main(void) {
  int board[LENGTH][LENGTH] = { 0,};
  char turn = 'X';
  int count = 0, row, column;

  /* #1. 보드 전체 출력  */
  while (1) {
    printf("Turn[%c] (row column): ", turn);
    scanf("%d %d", &row, &column);
    printf("\n");

    /* #2. 입력이 올바른지 확인
    올바르지 않다면 continue 사용하여 다시 입력받음*/

    /* #3. 보드의 입력받은 위치에 플레이어 표시 */
    if (turn == 'X') board[row][column] = 1;
    else board[row][column] = -1;

    /* #4. 보드 전체 출력 */
    /* #5 : 이겼는지 확인
          이겼으면 이긴 Player 출력하고 반복문 빠져나옴*/

    /* #6 : 총 횟수 증가 및 더 이상 둘 곳이 있는지 확인 */
    if (++count >= 9) {
      printf("Nobody wins!\n");
      break;
    }

    /* #7 : 플레이어 변경. X->O, O->X*/
    if (turn == 'X') turn = 'O';
    else turn = 'X';
  }
}
```

실습 4 – Step 2

- 보드에 플레이어를 잘 표시했는지를 보기 위해 보드 전체를 출력하는 함수를 만들어서 main()안에서 사용해보세요
  - #1, #4를 완성해주세요(@main)

```c
#include <stdio.h>
#define LENGTH 3

/*보드 전체 출력*/
void show_board(int board[][LENGTH])
{
    /* 채워보세요*/
    /* 보드 = 3x3 행렬*/
    /* 값이 1이면 'X', 값이 -1이면 '0'을 출력*/
    /* 값이 1, -1 둘다 아니면 ' ' 출력(Space)*/
}
```

```
./main
    0 1 2
    ------
0 |
1 |
2 |

Turn[X] (row column): 0 0

    0 1 2
    ------
0 |  X
1 |
2 |

Turn[O] (row column): 0 1

    0 1 2
    ------
0 |  X O
1 |
2 |
```

실습 4 – Step 2 – 풀이

```c
void show_board(int board[][LENGTH]) {
  char ch;
  printf("     0 1 2\n");
  printf("     ------\n");
  for (int i = 0; i < LENGTH; i++) {
    printf("%d | ", i);

    for (int j = 0; j < LENGTH; j++) {
      if (board[i][j] == 1)
        ch = 'X';
      else if (board[i][j] == -1)
        ch = 'O';
      else
        ch = ' ';
      printf(" %c", ch);
    }
    printf("\n");
  }
  printf("\n");
}
```

@main()

```c
int main(void)
{
  int board[LENGTH][LENGTH] = {0,};
  char turn = 'X';
  int count = 0, row, column;
  /* #1 : 보드 전체 출력*/
  show_board(board);

  while (1) {
    printf("Turn[%c] (row column): ", turn);
    scanf("%d %d", &row, &column);
    printf("\n");
    /* #2 : 입력이 올바른지 확인*/

    /* #3. 보드의 입력받은 위치에 플레이어 표시 */
    if (turn == 'X') board[row][column] = 1;
    else board[row][column] = -1;

    /* #4. 보드 전체 출력 */
    show_board(board);
```

실습 4 – Step 3

- 입력이 올바른지 확인하는 함수를 작성해보세요. 해당 함수를 사용해서 잘못된 입력을 받으면 다시 행렬을 입력받게 해보세요
  - #2 완성해주세요(@main)

```
int check_input(int board[][LENGTH],int row,int column){

    /*row, column이 0 ~ 2의 값을 가지는지?*/
    /*이미 체크된 곳인지?*/
    /*만약 잘못된 인풋을 받았다면 에러메세지도 출력해주세요*/
    /*입력이 올바르면 1 반환, 아니면 0 반환*/

     return 1;
}
```

```
Turn[X] (row column): -1 1

Invalid input. out of range
Turn[X] (row column): 3 1

Invalid input. out of range
Turn[X] (row column): 0 0

    0 1 2
    ------
0 |  X
1 |
2 |

Turn[O] (row column): 0 0

Invalid input. duplicate input
```

실습 4 – Step 3 – 풀이

```c
int check_input(int board[LENGTH][LENGTH], int row,
                                           int column) {
  /*row, column이 0 ~ 2의 값인지?*/
  if (row > 2 || row < 0 || column > 2 || column < 0) {
    printf("Invalid input. out of range\n");
    return 0;
  }

  /*이미 체크가 된 곳인지?*/
  if (board[row][column] != 0) {
    printf("Invalid input. duplicate input\n");
    return 0;
  }
  return 1;
}
```

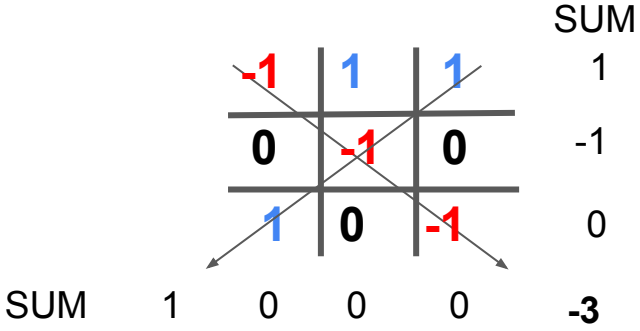
@main()

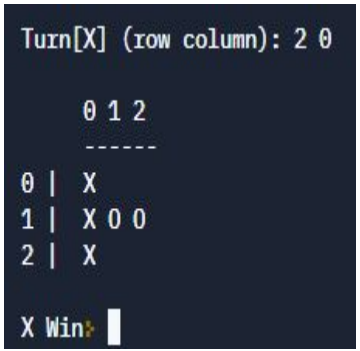
```c
int main(void)
{
  int board[LENGTH][LENGTH] = {0,};
  char turn = 'X';
  int count = 0, row, column;
  /* #1 : 보드 전체 출력*/
  show_board(board);

  while (1) {
    printf("Turn[%c] (row column): ", turn);
    scanf("%d %d", &row, &column);
    printf("\n");
    /* #2 : 입력이 올바른지 확인*/
    if (check_input(board, row, column) == 0)
      continue;
```

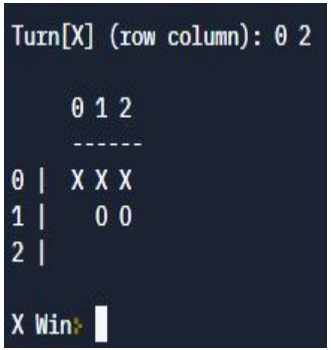
실습 4 – Step 4

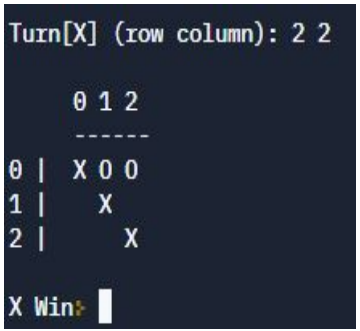
- 마지막으로 누군가가 이긴 것을 판단하는 함수를 만들어서 매번 플레이어가 수를 두고 이겼는지를 판단해보세요
  - #5를 완성해주세요

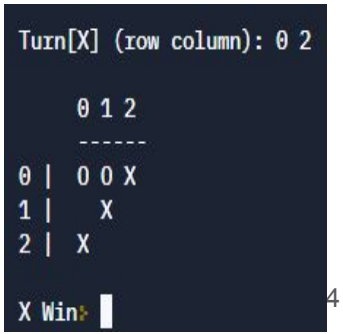
```
int check_win(int board[][LENGTH]) {
  /* 세로줄 3곳, 가로줄 3곳, 대각선 2곳 확인
     세개의 값을 더한 값이 3혹은 -3이면 누군가 이긴것임
     3이면 'X', -3이면 'O'가 이김
     입력이 올바르면 1 반환 아니면 0반환 */ }
```



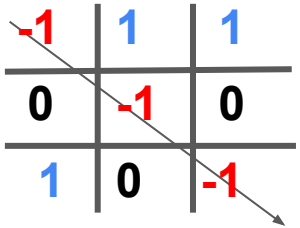








실습 4 – Step 4 – 풀이



board[0][0]+
board[1][1]+
board[2][2]
= -3

'O' win!!

```
int check_sum(int sum) {
  // 누군가 이김
  if (sum == 3 || sum == -3)
    return 1;
  // 이긴사람이 없음
  return 0;
}

int check_win(int board[][LENGTH]) {
  int sum;

  // 세로줄 확인
  for (int c = 0; c < LENGTH; c++) {
    sum = 0;
    for (int r = 0; r < LENGTH; r++)
      sum += board[r][c];
    if (check_sum(sum))
      return 1;
  }
  // 가로줄 확인
  for (int r = 0; r < LENGTH; r++) {
    sum = 0;
    for (int c = 0; c < LENGTH; c++)
      sum += board[r][c];
    if (check_sum(sum))
      return 1;
  }

  // 대각선 확인
  sum = 0;
  for (int r = 0; r < LENGTH; r++)
    sum += board[r][r];
  if (check_sum(sum))
    return 1;

  // 역대각선 확인
  sum = 0;
  for (int r = 0; r < LENGTH; r++)
    sum += board[r][LENGTH -1 -r];
  if (check_sum(sum))
    return 1;
  return 0;
}
```

실습 4 – Step 4 – 풀이

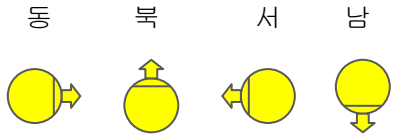
@main()

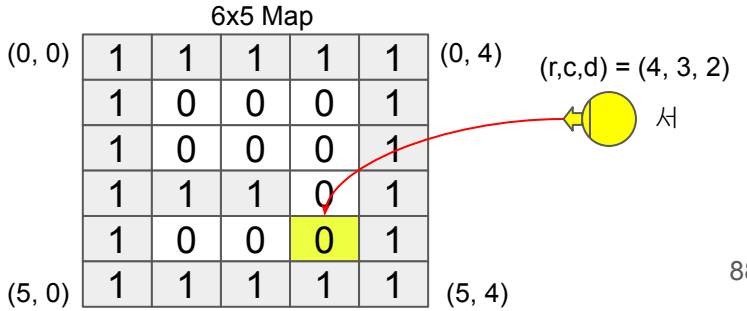
```
while (1) {
  /* ..... */

  /* #4. 보드 전체 출력 */
  show_board(board);

  /* #5 : 이겼는지 확인 */
  if (check_win(board)) {
    printf("%c Win", turn);
    break;
  }
```

과제 1

과제 1 – 설명

- 로봇 청소기가 움직인 거리 및 청소한 칸의 개수를 구하세요
  - N x M 크기의 Matrix형태의 Map을 가짐
  - 청소기는 한칸씩 이동을 하며 청소진행. 벽은 이동할 수 없음 (빈칸 : 0, 벽 : 1, 청소한 곳 : 2)
  - 청소기가 바라보는 방향은 동,서,남,북 중 하나임 (동 : 0, 북 : 1, 서 : 2, 남 : 3)
  - 지도의 북쪽으로부터 r번째, 서쪽으로부터 c번째 위치한 칸을 (r, c)라고 부름
  - 입력
    - 세로크기 : N, 가로크기 : M (3 <= N, M <= 10)
    - 청소기의 초기위치 (r, c), 바라보는 방향 d (동 : 0, 북 : 1, 서 : 2, 남 : 3)
    - 빈칸과 벽을 나타내는 NxM 행렬 Map (빈칸 : 0, 벽 : 1)
    - 지도의 첫 행, 마지막 행, 첫 열, 마지막 열은 모두 벽임
  - 출력
    - 움직인 거리, 청소한 칸의 개수
    - 청소 후 전체 맵

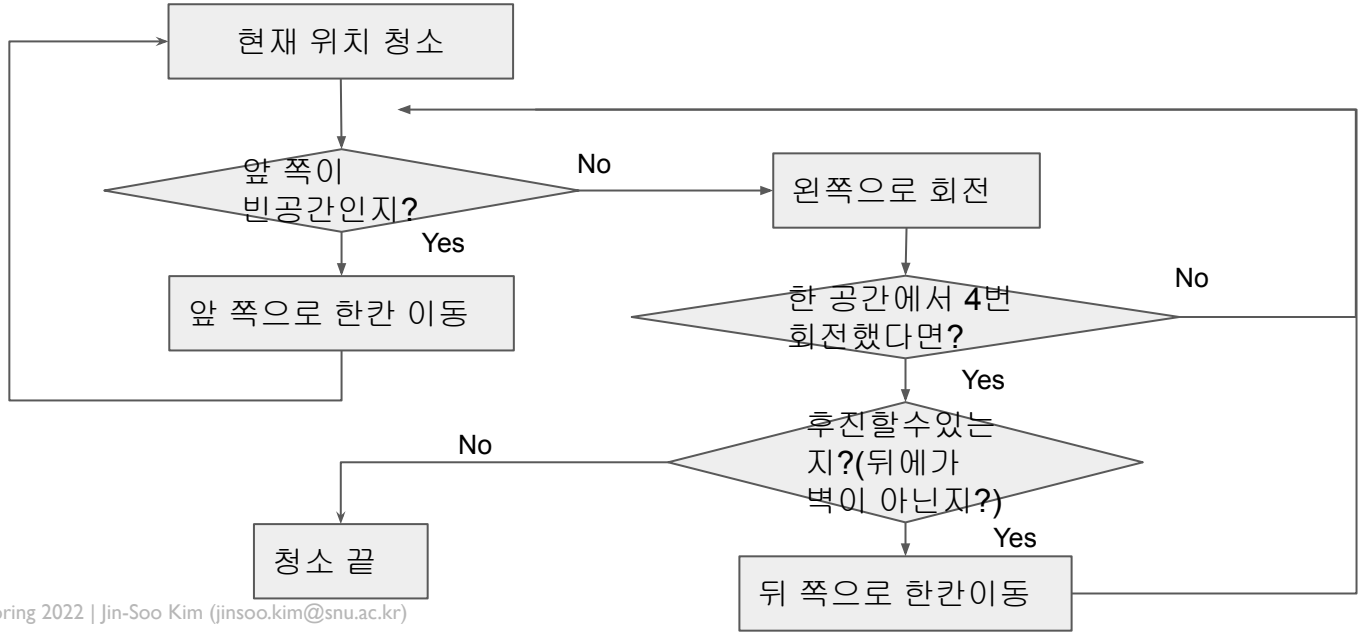




과제 1 – 설명

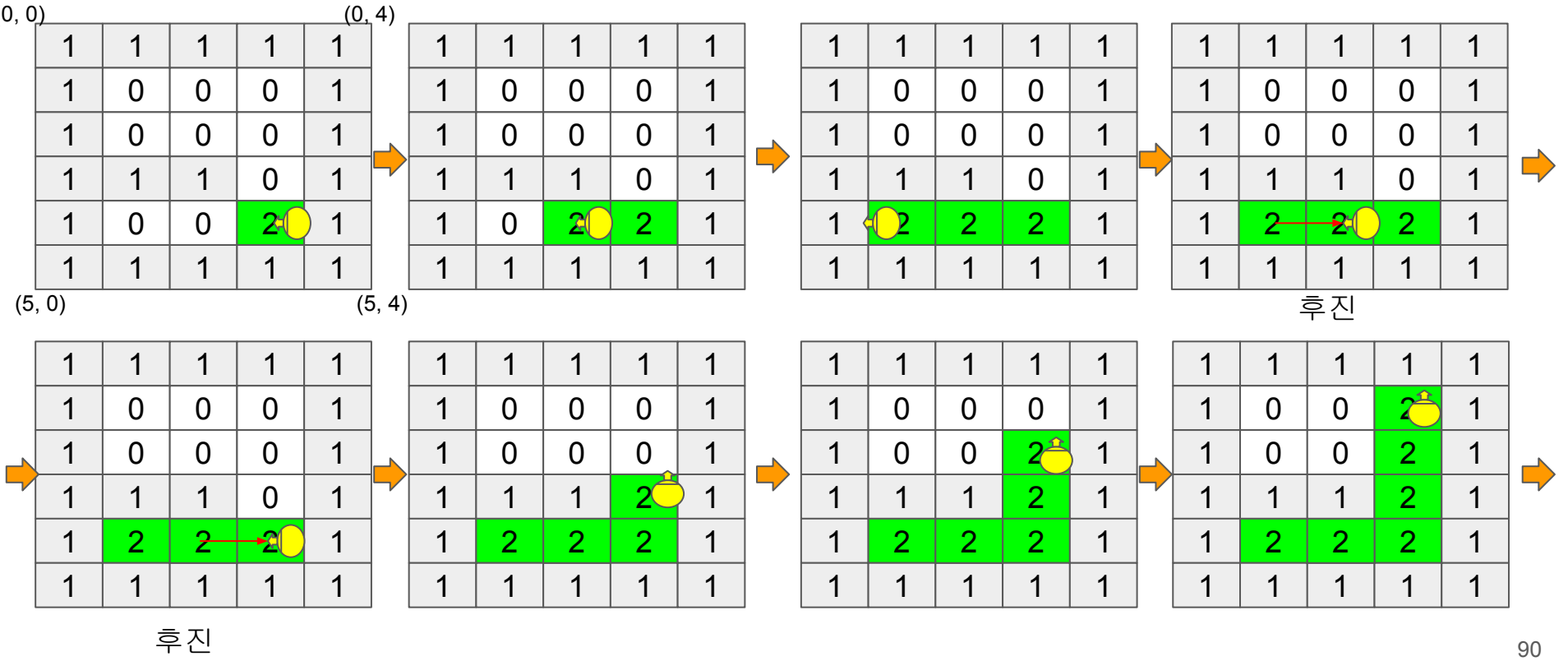
청소 순서

1. 현재 위치 청소함
2. 현재 방향의 앞쪽에 청소하지 않은 빈 공간이 존재한다면, 앞쪽으로 한칸 전진하고 1번으로 돌아감. 그렇지 않을 경우 왼쪽으로 회전하여 다시 2번 진행함
3. 2번 단계가 연속으로 4번 실행되었을 경우(현재 위치를 기준으로 동,서,남,북 모두가 벽 혹은 이미 청소한 곳이라면) 한 칸 후진하고 2번 다시 진행. 만약 바로 뒤쪽이 벽이라면 청소 끝



과제 1 – 설명

0 : 청소해야할 곳
1 : 벽
2 : 청소한 곳
초기 위치 : (4, 3, 2(서))



과제 1 – 설명

주변에 청소안한 칸은 없고,
뒷쪽은 벽이라 후진할 수
없으므로 청소 끝

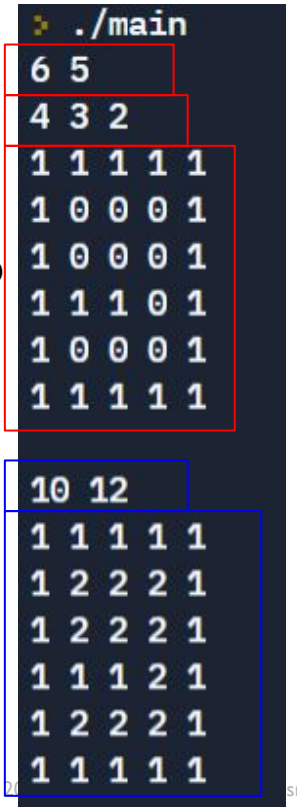
청소한 칸의 개수 : 10
움직인 거리 : 12

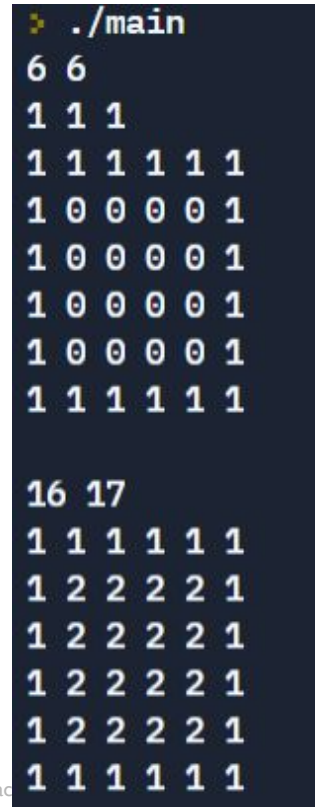
과제 1 – 출력 예시

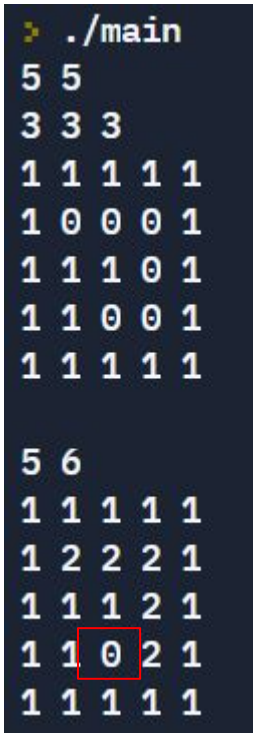
입력 1 : N, M
입력 2 : 초기 위치
(Row,Column, Direction)
(동: 0, 북:1, 서:2, 남:3)

입력 3: 벽을 포함한 Map
(1:벽, 0:청소해야할곳)

출력 1 :
청소한 칸의 개수,
이동한 횟수
출력 2 : Map 전체
(2: 청소를 한 곳)







청소를 못하는 부분도
발생함

과제 1 - 평가 기준

- 기준 1
  - 프로그램이 동작하는가? - 1점
- 기준 2
  - 프로그램이 정상적으로 입력을 받는가? - 1점
- 기준 3
  - 기준에 부합하여 결과를 출력하는가? - 3점

과제 2

과제 2 – 설명

- 주어진 영단어의 알파벳을 모두 입력할 때까지, 알파벳을 한 글자씩 입력하며 주어진 횟수안에 맞추는 게임을 만들어봅시다
  - 정답이 되는 영단어 하나를 입력하고 엔터키를 누름 (10글자 이하)
    - getchar()사용시 '\n'이 나올 때까지 입력
    - scanf("%s") 사용해도 됨
  - 정답을 맞추기 위해 한 글자씩 알파벳을 입력하고 엔터키를 누름
  - 알파벳을 하나씩 입력할 때마다 현재 상태를 화면에 출력
  - 영단어 길이의 2배까지 시도 가능
  - 단어를 맞추거나, 최대 시도 횟수를 경과하면 프로그램 종료
  - 대소문자는 구분하지 않음

```
➜ ./main
Apple
[0]: a
A _ _ _ _
[1]: b
A _ _ _ _
[2]: l
A _ _ l _
[3]: p
A p _ l _
[4]: p
A p p l _
[5]: e
A p p l e
Success
```

Tip. 한 글자씩 입력 받기 위해 getchar() 함수를 사용하고 나서 엔터 문자를 버퍼에서 제거해줘야 정상 동작함

과제 2 – 출력 예시

입력 1 : 정답 영단어
입력 2 : 알파벳

출력 1 : 현재상황.
맞춘 알파벳은 출력하고,
그렇지 않은 알파벳은 '_'
으로 출력
(알파벳 사이에 공백)

출력 2 : 제한된 횟수안에
정답을 맞추면 Success,
아니면 Fail 출력

```
./main
apple
[0]: p
_ p _ _ _
[1]: l
_ p _ l _
[2]: p
_ p p l _
[3]: p
_ p p l _
[4]: e
_ p p l e
[5]: a
a p p l e
Success
```

대소문자 구분안함
상태 출력시 정답의 것으로 출력

```
./main
ProGram
[0]: p
P _ _ _ _ _ _
[1]: R
P r _ _ _ _ _
[2]: O
P r o _ _ _ _
[3]: g
P r o G _ _ _
[4]: R
P r o G r _ _
[5]: A
P r o G r a _
[6]: M
P r o G r a m
Success
```

최대 횟수 : 단어길이 *2
BUS -> 6번(3*2)

```
./main
BUS
[0]: a
_ _ _
[1]: b
B _ _
[2]: c
B _ _
[3]: d
B _ _
[4]: e
B _ _
[5]: f
B _ _
Fail
```

과제 2 - 평가 기준

- 기준 1
  - 프로그램이 동작하는가? - 1점
- 기준 2
  - 프로그램이 정상적으로 입력을 받는가? - 1점
- 기준 3
  - 기준에 부합하여 결과를 출력하는가? - 3점