4190.103A-001: Programming Practice Lab. 7

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

Systems Software & Architecture Lab.

Seoul National University

Spring 2022

과제 풀이

Lab 6 과제 1

Lab 6 – 과제 1

- 지구 상에서 두 지점 거리 구하기
  - 앞서 설명한 math library를 이용하면 지구상 두 지점의 거리를 구할 수 있습니다.
  - 아래 “구면 코사인 법칙”을 이용해서 지구상의 두 지점간 거리를 구해보세요.

Radian 각도로 표현된 두 지점 사이 거리

$$(lat1, lon1), \quad (lat2, lon2)$$

$$\cos\theta = \sin(lat1)\sin(lat2) + \cos(lat1)\cos(lat2)\cos(lon2 - lon1)$$

$$\theta = \operatorname{acos}(\cos(\theta))$$

$$distance = r\theta$$

Lab 6 – 과제 1 – Tips

- 아래 template code의 function prototype을 활용해서 코드를 작성해보세요.

```c
#include <math.h>
#define pi 3.14159265358979323846
#define radius 6371

/* complete functions */
double deg2rad(double deg);
double distance(double lat1, double lon1, double lat2, double lon2);

int main(void)
{
  /* write your code */
}
```

Lab 6 – 과제 1 풀이

- degree -> radian 변환하는 함수
- 두 지점의 위도, 경도를 인자로 받아 거리를 반환하는 함수

```c
#include <stdio.h>
#include <math.h>
#define PI 3.14159265358979323846
#define RADIUS 6371

double deg2rad(double deg) {
  return deg * PI / 180;
}

double distance(double lat1, double lon1, double lat2, double lon2) {
  double cos_theta = sin(lat1) * sin(lat2) + cos(lat1) * cos(lat2) * cos(lon2 - lon1);
  double theta = acos(cos_theta);
  return RADIUS * theta;
}
```

$$\cos\theta = \sin(lat1)\sin(lat2) + \cos(lat1)\cos(lat2)\cos(lon2 - lon1)$$

$$\theta = acos(\cos(\theta))$$

$$distance = r\theta$$

Lab 6 – 과제 1 풀이

```c
int main(void) {

  double lat1, lon1, lat2, lon2;
  scanf("%lf %lf %lf %lf", &lat1, &lon1, &lat2, &lon2);

  lat1 = deg2rad(lat1); // lat1 Degree -> Radian
  lon1 = deg2rad(lon1); // lon1 Degree -> Radian
  lat2 = deg2rad(lat2); // lat2 Degree -> Radian
  lon2 = deg2rad(lon2); // lon2 Degree -> Radian

  printf("%.1lf\n", distance(lat1, lon1, lat2, lon2));

  return 0;
}
```

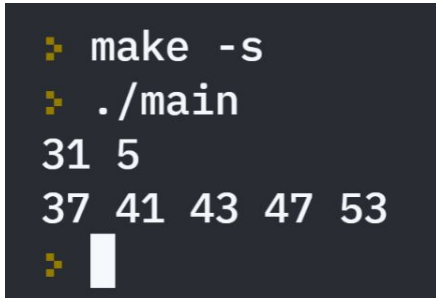
Lab 6 – 과제 1 학생답안

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

int main(void) {
  float lat1, lon1, lat2, lon2, dist, cs, theta;
  scanf("%f %f %f %f", &lat1, &lon1, &lat2, &lon2);
  cs = sin(3.141592654*lat1/180)*sin(3.141592654*lat2/180) +
cos(3.141592654*lat1/180)*cos(3.141592654*lat2/180)*cos(3.141592654*(lon2-lon1)/180);
  theta = acos(cs);
  dist = 6371*theta;
  printf("%.1f", dist);
  return 0;
}
```

Lab 6 과제 2

Lab6 – 과제 2

- 확장된 소수 판별 프로그램
  - 두 양의 정수 N, M을 입력 받아 N 보다 큰 소수를 작은 것부터 차례로 M개 출력하는 프로그램을 작성하시오
  - 입력: 두 개의 양의 정수 N, M (1<=N<=100, 1<=M<=5)
    - 예시: 31 5
  - 출력: N보다 큰 M개의 소수 – 각 숫자 사이는 space로 띄운다. 맨 뒤도 띄우고 개행함.
    - 예시: 37 41 43 47 53
  - 예전에 for, modular 연산으로 소수 판별을 연습 했습니다.
  - ※ 1과 그 수 자신 이외의 자연수로는 나눌 수 없는 자연수



Lab6 – 과제 2 풀이

- next_prime을 m번 호출하여 n보다 큰 소수 m개를 출력함

```c
int main(void) {
  int n, m;
  scanf("%d %d", &n, &m);

  for (int i = 0; i < m; i++) {
    n = next_prime(n);
    printf("%d ", n);
  }

  printf("\n");

  return 0;
}
```

Lab6 – 과제 2 풀이

- 소수인지 확인하는 함수 (int is_prime(int n))
  - 2부터 √n 까지 증가시키면서 n과 나누어 떨어지는 것이 있는지 확인
  - 나누어 떨어지는 것이 있으면 거짓, 아니면 참
  - double sqrt(double n)(@math.h)
    - 제곱근을 구하는 함수(√n)

- 다음 소수를 찾는 함수 (int next_prime(int n))
  - n을 증가시키면서 n보다 크고 가장 가까운 소수를 return함

```c
#include <stdio.h>
#include <math.h>
int is_prime(int n) {

  int root_n = (int)sqrt((double)n);

  for (int i = 2; i <= root_n; i++) {
    if (n % i == 0)
      return 0;
  }

  return 1;
}

int next_prime(int n) {
   while (1) {
    n++;
    if (is_prime(n))
      return n;
  }
}
```

Lab6 – 과제 2 학생 답안

```c
int main(void) {
  int n, m;
  scanf("%d %d", &n, &m);

  while(1) {
    n++;
    for (int i = 2; i <= n; i++) {
      if (i == n) {

        printf("%d ", n);
        m--;
        break;
      }
      else if (n % i == 0) break;
    }
    if (m == 0) break;
  }

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

Lab6 – 과제 2 학생 답안

```c
int isPrime(int n){
  for(int i=2;i<n-1;i++){
    if(n%i==0)
      return 0;
  }
  return 1;
}
```

```c
int main(void) {
  int a, b;
  scanf("%d %d", &a, &b);
  int num=0;
  a=a+1;
  while(num!=b){
    if(isPrime(a)==1){
      printf("%d ", a);
      num++;
    }
    a++;
  }

  return 0;
}
```

실습 1

실습 1 – 설명

- Switch 문
  - 변수와 동일한 값을 가지는 case :로 이동하여 실행문을 순차적으로 실행
  - 변수와 동일한 값을 가지는 case :가 없다면 default : 로 이동하여 순차적으로 실행
  - break 문을 만나면 switch를 빠져나감, break가 없다면 아래 구문들을 순차적으로 실행

```
switch(변수){
    case 값1 :                          // if (변수 == 값1) { }
        실행문1;
        break; //switch 문을 빠져나감
    case 값2 :                          // else if (변수 == 값2) { }
        실행문2;
        break;
    default :                           // else { }
        실행문3;
}
```

실습 1 – 설명

- 점수 –> 학점 변환 프로그램
  - A : 90점 이상 , B : 80이상, C : 70이상, D : 60점이상, 나머지 : Fail

```c
int main(void) {
  int n;
  printf("What is your score?: ");
  scanf("%d", &n);

  switch(n/10){
    case 10:
    case 9 :
      printf("A\n");
      break;

    case 8 :
      printf("B\n");
      break;

    case 7 :
      printf("C\n");
      break;

    case 6 :
      printf("D\n");
      break;

     default :
      printf("Fail\n");
  }
  return 0;
}
```

```
make -s
./main
What is your score?: 100
A
./main
What is your score?: 85
B
./main
What is your score?: 50
Fail
```

* break도 제거하고 실행하여 결과를 확인해보세요

실습 1 – 설명

- 학점 –> 점수 변환 프로그램
  - A : 90점 이상 , B : 80이상, C : 70이상, D : 60점이상, F : 60점 미만

```c
int main(void) {
  char grade;
  printf("What is your grade?: ");
  scanf("%c", &grade);

  switch(grade){
    case 'A':
      printf("90+\n");
      break;

    case 'B' :
      printf("80+\n");
      break;

    case 'C' :
      printf("70+\n");
      break;
    case 'D' :
      printf("60+\n");
      break;
    case 'F' :
      printf("60-\n");
      break;
    default :
      printf("Invalid grade\n");
  }
  return 0;
}
```

```
./main
What is your grade?: A
90+
./main
What is your grade?: C
70+
./main
What is your grade?: F
60-
./main
What is your grade?: G
Invalid grade
```

실습 1 – 제출

- 두개의 피연산자(정수)와 하나의 연산자를 입력받아 계산 값을 출력하는 프로그램 작성
  - 입력: 피연산자1(int) 연산자(char) 피연산자2(int)
  - 출력: 계산 값 (int)
  - 연산자 : +, −, *, x, /, %, <, >
  - *, x 연산자는 곱하기를 의미함
  - 지원하지 않는 연산자가 입력될 경우 “Invaild operator” 출력
  - 입력값과 출력값이 int형의 범위를 넘어서는 값은 고려하지 않음
  - Switch문을 사용

```
./main
10 + 20
10 + 20 = 30
./main
10 * 5
10 * 5 = 50
./main
5 * 10
5 * 10 = 50
./main
10 / 2
10 / 2 = 5
./main
10 % 3
10 % 3 = 1
./main
10 < 1
10 < 1 = 0
./main
10 > 1
10 > 1 = 1
./main
10 @ 1
Invalid operator
```

실습 1 – 제출

```c
int main(void) {
  int operand1;
  int operand2;
  char op;
  int result;
  scanf("%d %c %d", &operand1, &op, &operand2);

  switch (op)
  {
    case '+' :
      result = operand1 + operand2;
      break;
    case '-' :
      result = operand1 - operand2;
      break;
    case '*' :
    case 'x' :
    case 'X' :
      result = operand1 * operand2;
      break;
    case '/' :
      result = operand1 / operand2;
      break;
    case '%' :
      result = operand1 % operand2;
      break;
    case '>' :
      result = operand1 > operand2;
      break;
    case '<' :
      result = operand1 < operand2;
      break;

    default :
      printf("Invalid operator\n");
      return 0;
  }

  printf("%d %c %d = %d\n", operand1, op,
operand2, result);
  return 0;
}
```

```
> ./main
10 + 20
10 + 20 = 30
> ./main
10 * 5
10 * 5 = 50
> ./main
5 * 10
5 * 10 = 50
> ./main
10 / 2
10 / 2 = 5
> ./main
10 % 3
10 % 3 = 1
> ./main
10 < 1
10 < 1 = 0
> ./main
10 > 1
10 > 1 = 1
> ./main
10 @ 1
Invalid operator
```

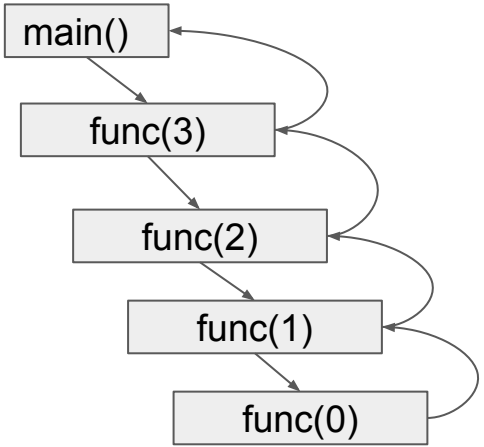
실습 2

실습 2 – 설명

- 재귀함수 : 함수 내 자기 자신을 호출하는 함수

```c
void func(int n){
    printf("Recursive call start  %d\n", n);
    if (n <= 0) { // 탈출조건
        printf("Recursive call done  %d\n", n);
        return;
    }

    func(n-1);
    printf("Recursive call done  %d\n", n);
}

int main(void){
    int n;
    scanf("%d", &n);
    func(n);
    return 0;
}
```

```
➜ ./main
3
Recursive call start 3
Recursive call start 2
Recursive call start 1
Recursive call start 0
Recursive call done 0
Recursive call done 1
Recursive call done 2
Recursive call done 3
```



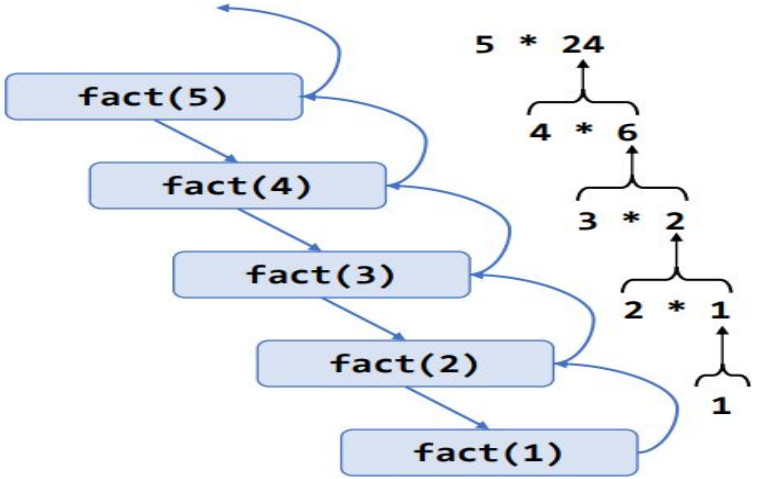
실습 2 – 설명

- 재귀함수를 이용한 Factorial 계산 프로그램

```c
int factorial(int n)
{
    if (n <= 1)
        return 1;
    else
        return n * factorial(n-1);
}

int main(void)
{
    int n, f;
    scanf("%d", &n);
    f = factorial(n);
    printf("%d! = %d\n", n, f);
    return 0;
}
```

$$f(n) = \begin{cases} n \times f(n-1) & \ldots. n \geq 1 \\ 1 & \ldots. n = 0 \end{cases}$$

```
./main
5
5! = 120
./main
4
4! = 24
```



실습 2 – 제출

- 재귀함수를 이용하여 최대공약수(GCD)를 구하는 유클리드 호제법을 구현해보세요
  - 유클리드 알고리즘
    - 1. 두 수 a, b(a > b)가 주어졌을때 a와 b의 나머지 연산 진행 (a % b = r)
    - 2. r이 0이라면, b가 최대공약수(GCD)
    - 3. r이 0이 아니라면, a에 b를 넣고, b에 r를 넣은 후 1번부터 다시 시작

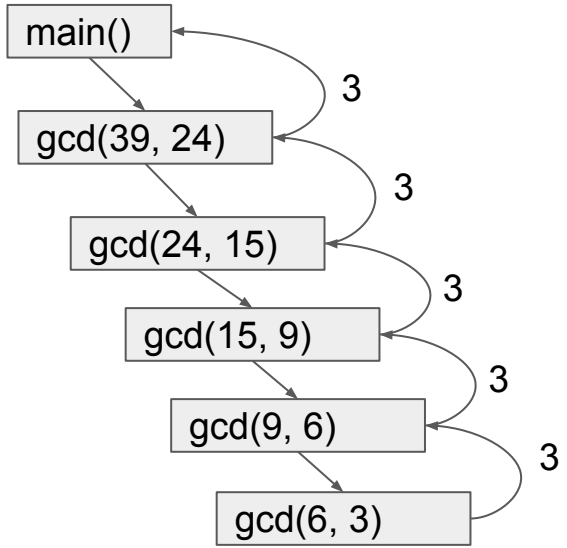
a = 39, b = 24
39 % 24 = 15
24 % 15 = 9
15 % 9 = 6
9 % 6 = 3
6 % **3** = 0

GCD(39, 24) = 3

```c
int gcd(int a, int b)
{
  // write your code
}
int main(void) {
  int a, b;
  scanf("%d %d", &a, &b);
  printf("gcd(%d,%d) = %d\n", a, b, gcd(a,b));
  return 0;
}
```

실습 2 – 제출

```c
int gcd(int a, int b)
{
  int r, tmp;
  // b의 값이 a의 값보다 크다면 swap
  if (a < b) {
    tmp = a;
    a = b;
    b = tmp;
  }

  r = a%b;
  if (r == 0) //나머지가 0이면 b가 최대공약수
    return b;
  else
    return gcd(b, r);
}
```



실습 3

실습 3 – 설명

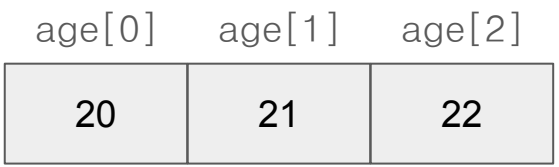
- 배열(array)
  - 동일한 타입의 데이터를 여러개 할당하여 담을 수 있는 기능
  - Index를 통해 각 데이터에 접근함
- 배열의 선언
  - 자료형 이름[크기]; 크기에 들어가는 값은 상수만 가능
  - int age[5]; //5개의 int형 데이터를 가지는 배열
  - double height[3]; // 3개의 double형 데이터를 가지는 배열
  - double weight[a]; // 컴파일에러

age[0] (int)	age[1] (int)	age[2] (int)	age[3] (int)	age[4] (int)

height[0] (double)	height[1] (double)	height[2] (double)

실습 3 – 설명

- 배열 접근
  - 배열의 index는 0부터 시작함
  - int age[3];
  - age[0] = 20; // age 배열 첫번째 element에 20 값을 넣음
  - int i = 1;
  - age[i] = 21; // index에 정수 형태 변수를 넣는 것도 가능
  - age[i+1] = 22;
  - age[i+2] = 23; // 할당하지 않은 메모리를 접근하기 때문에 문제 발생. 주의필요



실습 3 – 설명

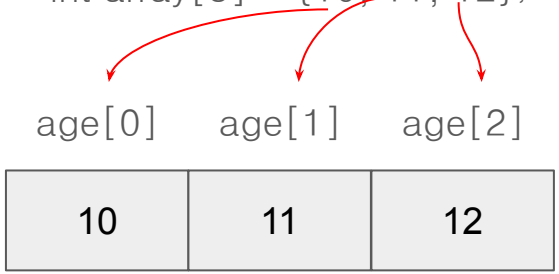
- 다음 코드를 작성해보고 배열을 익혀보세요

```c
int main(void) {
    int array[3];
    int n;
    array[0] = 10;
    array[1] = 11;
    array[2] = 12;
    printf("%d %d %d\n", array[0], array[1], array[2]);

    scanf("%d", &n);
    for (int i = 0; i < 3; i++)
    {
        array[i] = n++;
        printf("%d ", array[i]);
    }
    return 0;
}
```

```
> make -s
> ./main
10 11 12
100
100 101 102
>
> ./main
10 11 12
200
200 201 202
```

실습 3 – 설명

- 배열 초기화
  - int array[3] = {10, 11, 12};




```c
int main(void)
{
    int array[3] = {10, 11, 12};
    int array2[3] = {10, }; // 나머지는 0으로 초기화
    int array3[3]; // 초기화안함

    for (int i = 0; i < 3; i++)
    {
        printf("[%d] array: %d array2: %d array3: %d\n",
                    i, array[i], array2[i], array3[i]);
    }

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

```
./main
[0] array: 10 array2: 10 array3: -1194784800
[1] array: 11 array2: 0 array3: 1793901416
[2] array: 12 array2: 0 array3: 1793901416
```

실습 3

- 주사위를 n번 던져 각 면이 나오는 횟수 출력하는 프로그램
  - 입력
    - 횟수 n
  - 출력
    - 각 면(0 ~ 5)이 나온 횟수
  - n을 작은 값부터 충분히 큰 값까지 변경하며 실행하여 값들이 얼마나 random하게 분포하는지 확인해보세요
  - random 함수를 이용해주세요
    - srand(time(NULL));
    - rand() % 6;

```
> ./main
10
[0] 1
[1] 0
[2] 2
[3] 1
[4] 2
[5] 4
>
> ./main
100
[0] 14
[1] 21
[2] 16
[3] 19
[4] 11
[5] 19
>
> ./main
1000000
[0] 167355
[1] 166262
[2] 166423
[3] 167320
[4] 166290
[5] 166350
```

실습 3

```c
#include <stdio.h>
#define DICE_NUM  6

int main(void)
{
  int n;
  int count[DICE_NUM] = {0,}; // 0으로 초기화

  srand(time(NULL));
  scanf("%d", &n);

  for(int i = 0; i < n; i++)
    count[rand() % DICE_NUM]++; // 횟수증가


  for(int i = 0; i < DICE_NUM; i++)
    printf("[%d] %d\n", i, count[i]);

  return 0;
}
```

```
> ./main
10
[0] 1
[1] 0
[2] 2
[3] 1
[4] 2
[5] 4
>
> ./main
100
[0] 14
[1] 21
[2] 16
[3] 19
[4] 11
[5] 19
>
> ./main
1000000
[0] 167355
[1] 166262
[2] 166423
[3] 167320
[4] 166290
[5] 166350
```

실습 4

실습 4

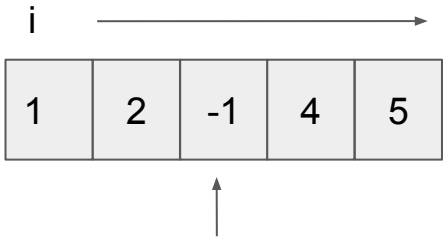
- 5개 정수의 통계를 내는 프로그램을 작성해보세요
  - 입력
    - 5개의 정수(int)
    - 허용 범위 : 0 <= n <= 100
    - 허용범위를 벗어난 정수 (n < 0 or n > 100)는 하나만 허용
    - 허용범위를 벗어난 정수가 두 개 이상인 경우 프로그램 중단
    - 허용범위를 벗어난 정수가 한 개의 경우 무효한 값이라고 판단하고
    - 나머지 4개의 수의 평균값(소수점아래는 버림)으로 대체함
  - 출력
    - 총 합(int), 평균(double), 최대값(int), 최소값(int), 보정된 값(int)
    - 평균 출력시 소수점 두번째자리까지 출력
  - 배열을 사용해보세요

```
> ./main
1 2 3 4 5
Sum: 15
Mean: 3.00
Max: 5
Min: 1
1 2 3 4 5
>
> ./main
1 2 -1  4 5
Sum: 15
Mean: 3.00
Max: 5
Min: 1
1 2 3 4 5
>
> ./main
1 2 -1 101 4
Invalid input
```

실습 4

```c
#include <stdio.h>

#define ARRAY_SIZE 5
#define MAX 100
#define MIN 0
int main(void) {

  int arr[ARRAY_SIZE];
  int sum = 0;
  double mean;
  int max = MIN;
  int min = MAX;
  int invalid_num = 0;
  int invalid_index;

  // 정수 입력
  for (int i = 0; i < ARRAY_SIZE; i++)
    scanf("%d", &arr[i]);
```

```c
  for (int i = 0; i < ARRAY_SIZE; i++)
  {
      // 허용 범위를 충족하는 경우
      if (arr[i] >= MIN && arr[i] <= MAX)
      {
        sum += arr[i];

        if (min > arr[i])
          min = arr[i];

        if (max < arr[i])
          max = arr[i];
      }
      else { // 허용 범위를 벗어날 경우
        invalid_num++;
        invalid_index = i;
      }
  }
```

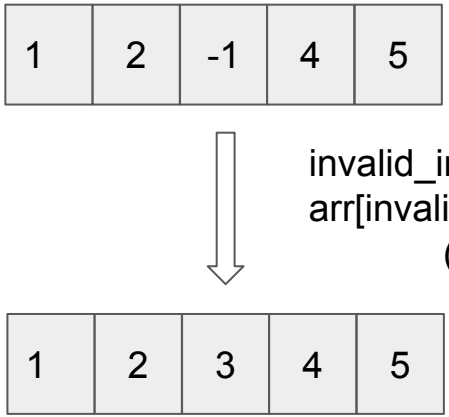


invalid_num = 1
invalid_index = 2

sum = 1 + 2 + 4 + 5
min = 1
max = 5

실습 4

```
//허용범위를 벗어나는 정수가 2개 이상
if (invalid_num > 1) {
  printf("Invalid input\n");
  return 0;
}
//허용범위를 벗어나는 정수가 1개 일때
else if (invalid_num > 0) {
  arr[invalid_index] = sum /(ARRAY_SIZE - 1);
  sum += arr[invalid_index];

  if (min > arr[invalid_index])
    min = arr[invalid_index];

  if (max < arr[invalid_index])
    max = arr[invalid_index];
}

mean = (double)sum/ARRAY_SIZE;
```



sum = 12 (1 + 2 + 4 + 5) + 3(보정된 값)
min = 1
max = 5
mean = 3 (sum / 5)

실습 4

```
  printf("Sum: %d\n", sum);
  printf("Mean: %.2lf\n", mean);
  printf("Max: %d\n", max);
  printf("Min: %d\n", min);

  for (int i = 0; i < ARRAY_SIZE; i++)
  {
    printf("%d ", arr[i]);
  }

  printf("\n");

  return 0;
}
```

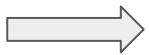
```
> ./main
1 2 -1 4 5
Sum: 15
Mean: 3.00
Max: 5
Min: 1
1 2 3 4 5
```

실습 4 – 제출

- 10개의 정수를 입력받고 각 정수가 몇 번 등장했는지, 정수의 숫자가 큰 수부터 부터(내림차순)으로 출력하세요
  - 입력
    - 10개의 정수(int) 입력
  - 출력
    - 등장한 정수와 그 개수를 출력 [정수 횟수]
    - 정수의 크기가 큰 수 부터 출력

입력

```
3 3 1 1 1 7 5 7 6 6
```



출력

```
7 2 // 7이 2번 등장
6 2
5 1 // 5가 1번 등장
3 2
1 3
```

실습 4 – 출력 예시

- 출력예시

입력

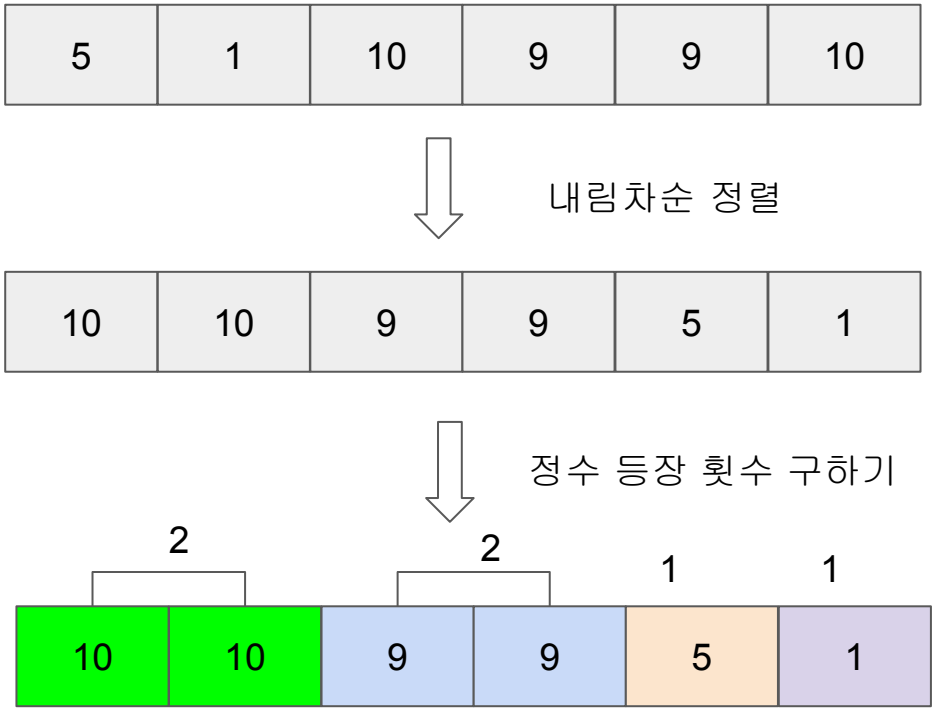
출력

```
> ./main
6 7 8 9 10 10 9 8 7 6
10 2
9 2
8 2
7 2
6 2
```

```
> ./main
10 10 10 10 10 10 10 10 10 1
10 9
1 1
```

```
> ./main
10 20 30 40 50 1 2 3 4 5
50 1
40 1
30 1
20 1
10 1
5 1
4 1
3 1
2 1
1 1
```

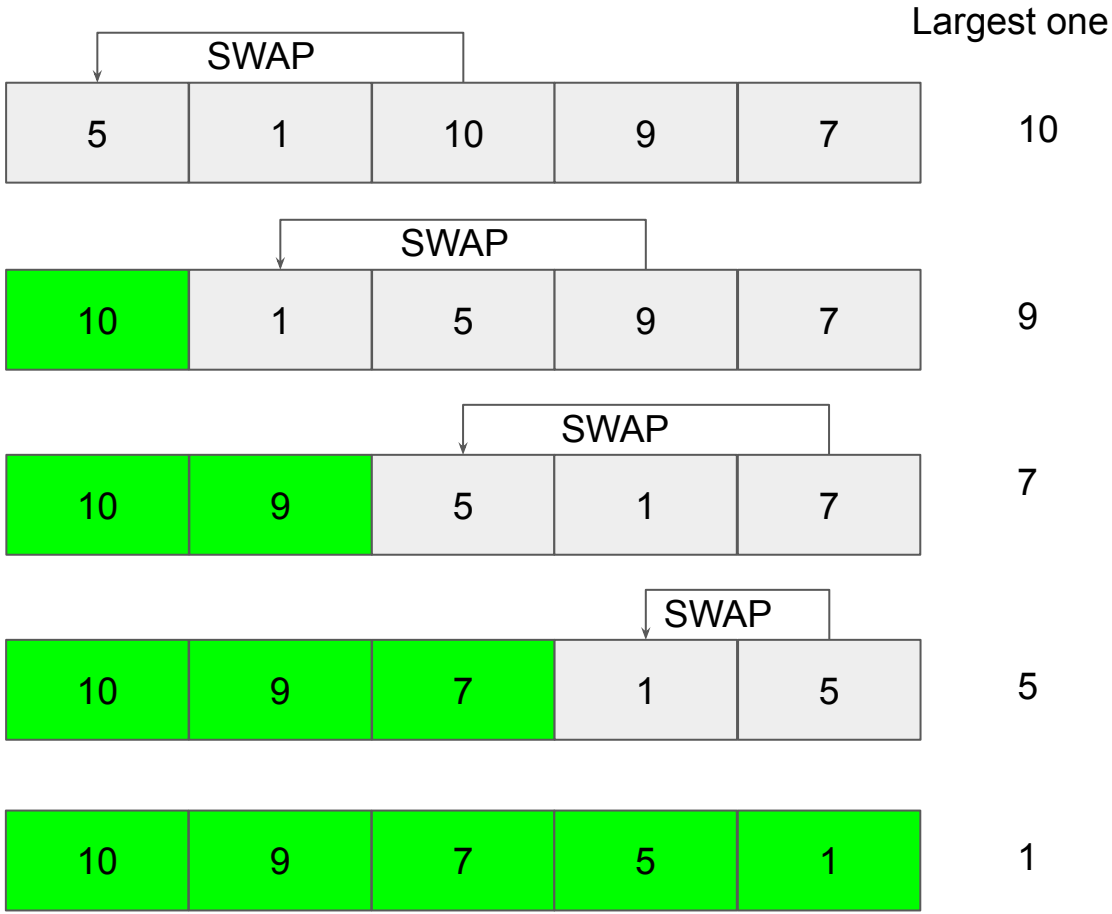
실습 4 – Tip




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

int main(void) {

  int arr[ARRAY_SIZE];

  // Step 1 : 정수 입력
  for (int i = 0; i < ARRAY_SIZE; i++)
    scanf("%d",&arr[i]);

  // Step 2 : 내림차순 정렬

  // Step 3 : 정수 등장 횟수 구하기

  // Step 4 : 결과 출력

  return 0;
}
```

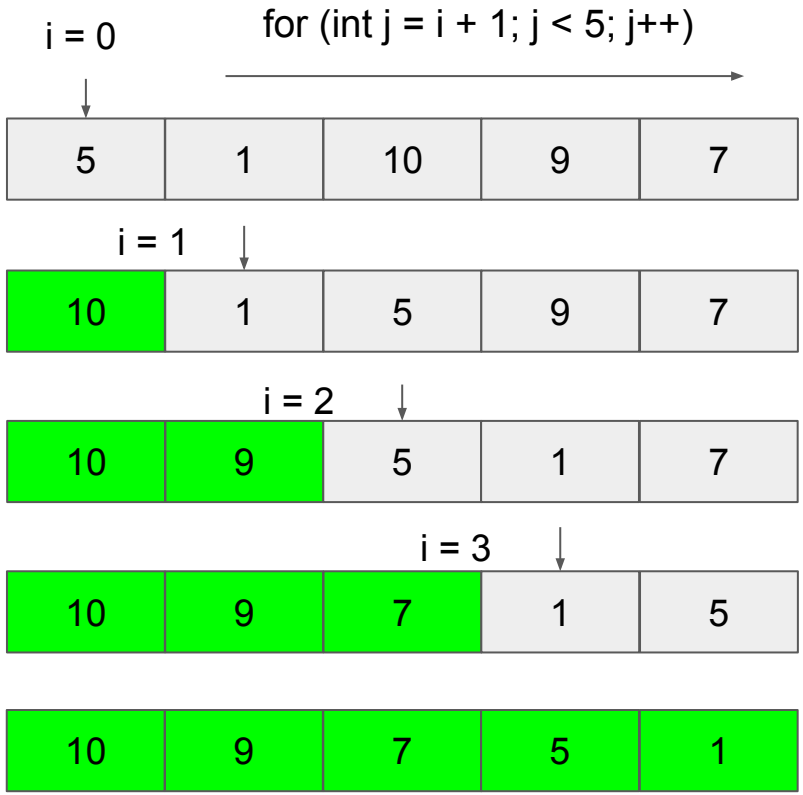
실습 4

- 정렬에는 많은 방법이 있습니다.
- 다음은 그 중 한가지 방법입니다.
  - selection sort



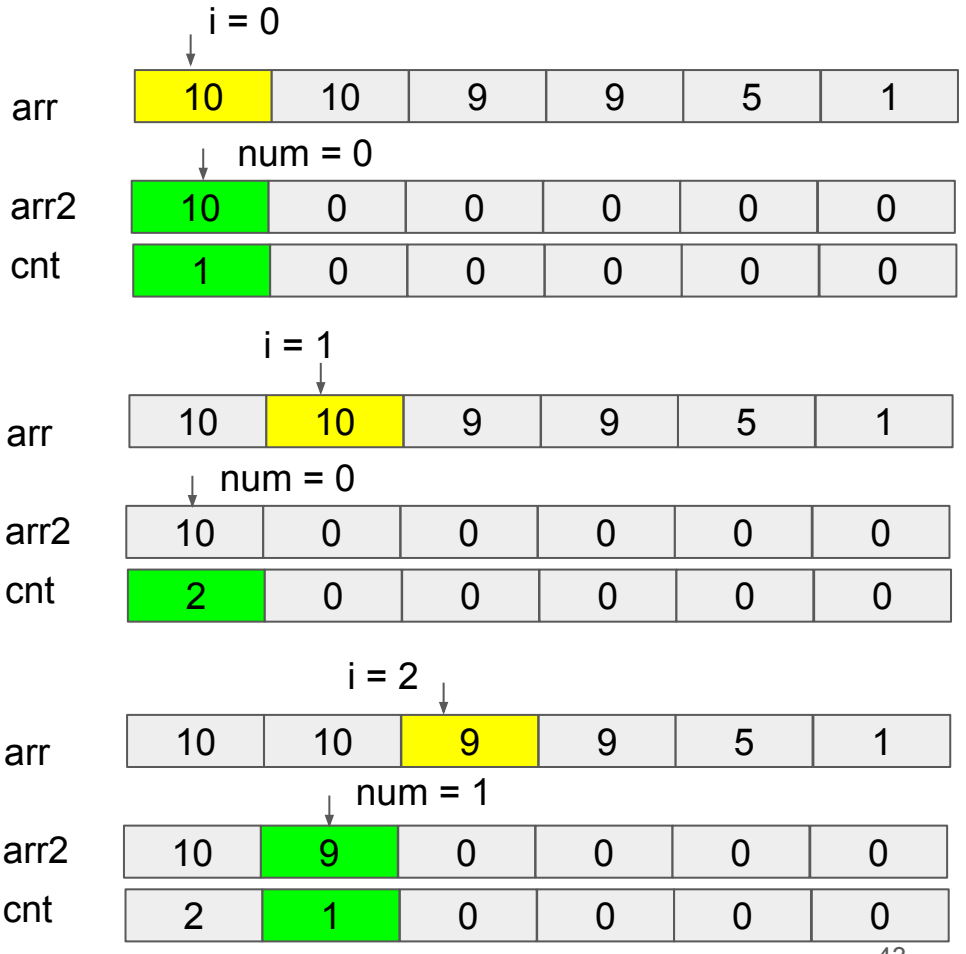
실습 4

```c
#define ARRAY_SIZE 10
int main(void) {

  int arr[ARRAY_SIZE];
  int arr2[ARRAY_SIZE];
  int cnt[ARRAY_SIZE] = {0,};
  int num = 0;
  int max_index, temp;

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

  // 가장 마지막 숫자 자동정렬
  for (int i = 0; i < ARRAY_SIZE-1; i++) {
    max_index = i;

    // 최대값 탐색
    for (int j = i + 1; j < ARRAY_SIZE; j++){
      if (arr[max_index] < arr[j])
        max_index = j;
    }
     // 최소값이 자기 자신이라면 그대로둠
    if (i != max_index) {
      // swap
      temp = arr[max_index];
      arr[max_index ] = arr[i];
      arr[i] = temp;
    }
  }
```



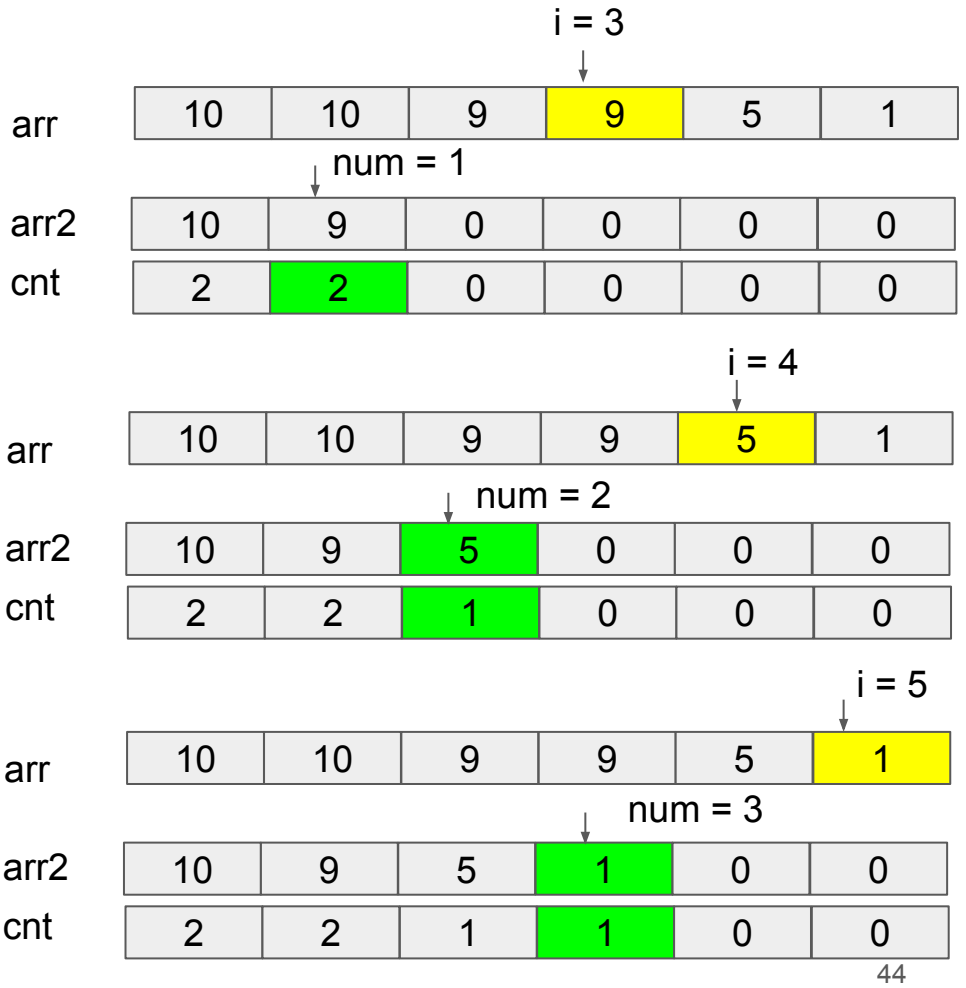
실습 4

```
  num = 0;
  arr2[num] = arr[0];
  cnt[num]++;

  for (int i = 1; i < ARRAY_SIZE; i++) {
    // 전 element와 값이 같다면 이미 등장한 정수
    if (arr[i] == arr[i-1]){
      cnt[num]++;
    }
    else {
      num++;
      arr2[num] = arr[i];
      cnt[num]++;
    }
  }

  for (int i = 0; i < ARRAY_SIZE; i++){
    if (cnt[i] > 0)
      printf("%d %d\n", arr2[i], cnt[i]);
  }
  return 0;
}
```



실습 4

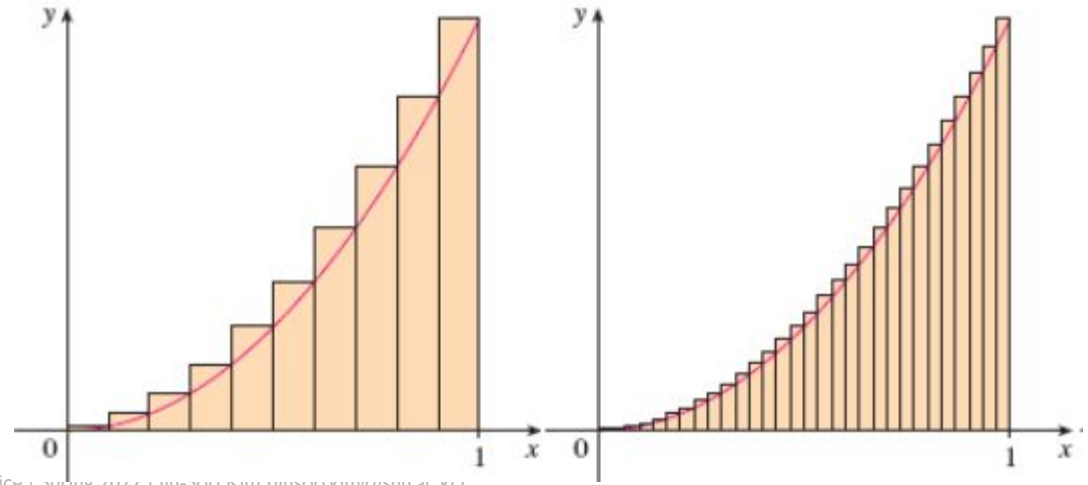
```
  num = 0;
  arr2[num] = arr[0];
  cnt[num]++;

  for (int i = 1; i < ARRAY_SIZE; i++) {
    // 전 element와 값이 같다면 이미 등장한 정수
    if (arr[i] == arr[i-1]){
      cnt[num]++;
    }
    else {
      num++;
      arr2[num] = arr[i];
      cnt[num]++;
    }
  }

  for (int i = 0; i < ARRAY_SIZE; i++){
    if (cnt[i] > 0)
      printf("%d %d\n", arr2[i], cnt[i]);
  }
  return 0;
}
```



과제 1

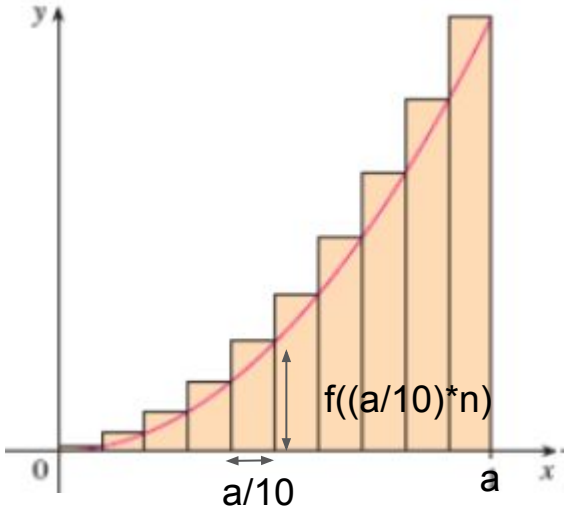
과제 1 – 문제

- 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
=> $(a^3/3) + (a^2/2)$

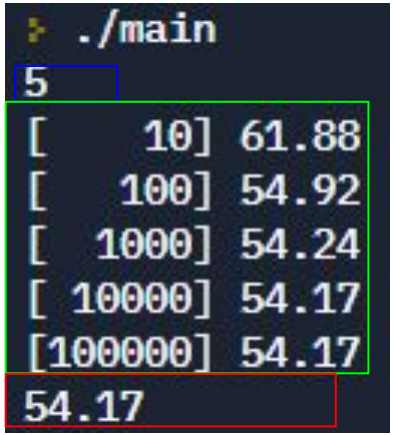
구분구적법
=> 사각형 밑변: a/10,
=> 사각형 높이: f((a/10)*n) , n은 1부터 시작
=> f(a/10)*a/10 + f(a/10*2)*a/10…. +f(a) *a/10
=> f(a/100)*a/100 + f(a/100*2)*a/100 … +f(a)*a/100

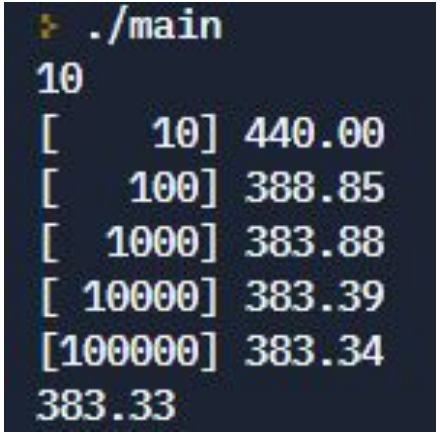
과제 1 – 문제 출력 예시

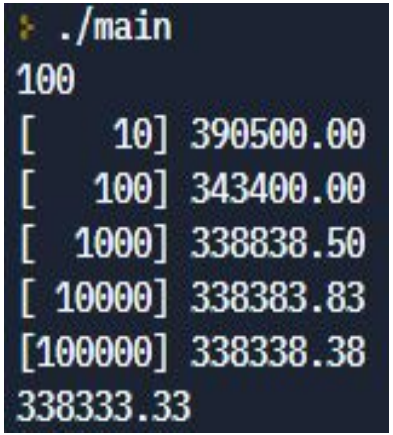
입력

출력 1
구분구적법을 통한 넓이
사각형 개수 10 ~ 100000

출력 2
적분공식을 이용한 넓이







과제 2

과제 2 – 문제

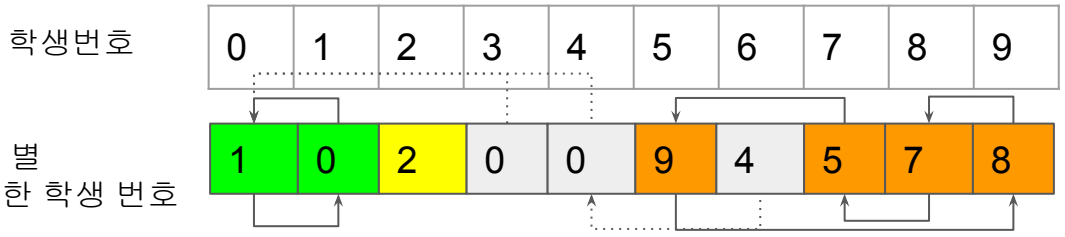
- 아래 조건을 만족하는 프로젝트팀을 구성하는 프로그램을 만들어보세요.
  - 조건
    - 학생은 10명이 있음. 각 학생의 번호는 (0 ~ 9)
    - 학생 한명당 함께하고 싶은 학생을 한명 선택함. 자기 자신을 선택해도 무방
    - 팀원 수에는 제한이 없음
    - 한 팀에 한 명만 있을 수도 있음 (자기 자신을 선택할 경우)
    - 한 팀이 되는 조건 : S1-> S2, S2 -> S3, … Sn -> S1 (최종적으로 다시 자신이 선택되어져야함)
  - 입력
    - 각 학생마다 선택한 학생의 번호를 뜻하는 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)



과제 2 – 문제 출력 예시

입력 : 각 학생 별 선택한 학생번호

출력 1
팀 개수

출력 2
각 팀별 학생번호
*번호가 작은 학생이 포함된 팀부터 출력
*학생번호는 작은 번호부터 출력