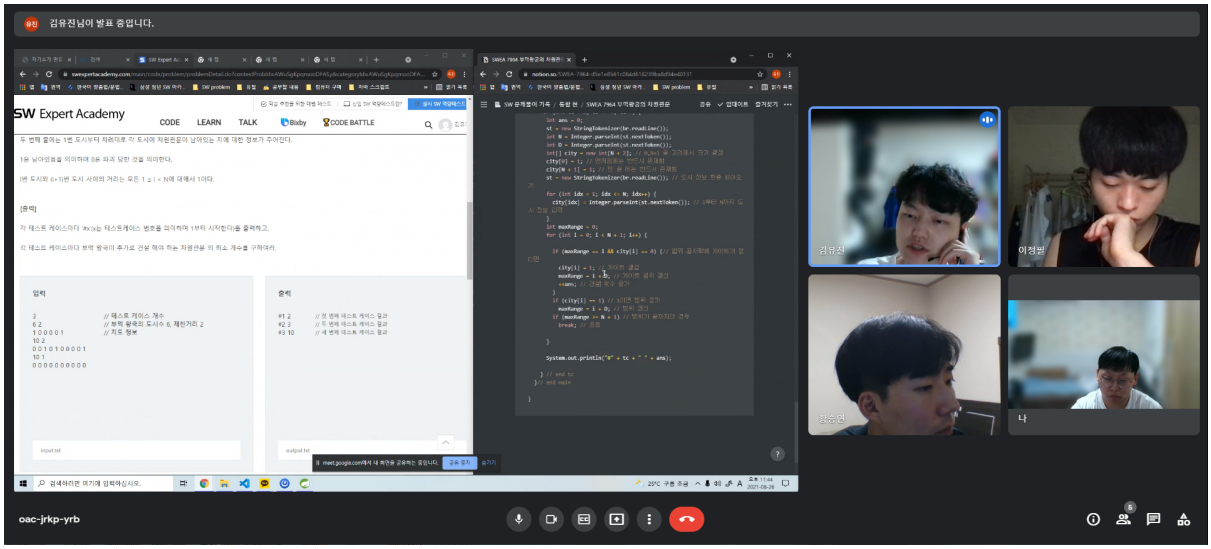
2021년 8월 26일 4주차 발표

조원 : 김유진, 오윤기, 황승연, 이정필 8월 26일 23시 발표



발표자 : 이정필 \ 문제 : 백준 2309 일곱난쟁이

9명의 난쟁이 중에서 7명을 선택했을 때 키의 합이 100인 경우를 출력하는 문제

단 답이 여러개인 경우가 있다. 그 중에 아무거나 출력하면 된다.

9 Combination 7이기 때문에 경우의 수가 36 밖에 안 나온다.

다 돌린 후에 마지막 정답을 출력, 처음에 정답이 나오면 출력 모두 상관없다.

다만 처음에 정답이 나오면 멈추는 동작을 하고 싶어서 다른 조원들이 각자의 생각을 이야기했다.

승연님의 경우 flag를 이용해서 재귀의 맨위에서 return하는 조건문을 이야기 했으며,

유진님의 경우 답이 맞는 경우 true를, 아닌 경우에는 false를 리턴해서 재귀가 수행되는 combi(cnt + 1, i + 1); 의 결과를 이용하여 다시 return 하는 구조를 설명하였다.

```
package algo;

import java.io.*;
import java.util.*;

public class BJ2309 {
    static int[] shorts = new int[9];
    static boolean[] isSelected = new boolean[9];
    static int[] sevenGGoma = new int[7];
    static boolean flag = false;
```

```
    public static void main(String[] args) throws Exception {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        for (int i = 0; i < 9; i++) {
            shorts[i] = Integer.parseInt(br.readLine());
        }

        Arrays.sort(shorts);

        combi(0, 0);
        flag = false;
    }

    private static void combi(int cnt, int start) {

        if (cnt == 7) {
            int sum = 0;
            for (int i = 0; i < 7; i++) {
                sum += sevenGGoma[i];
            }
            if (sum == 100 && flag == false) {
                for (int b : sevenGGoma) {
                    System.out.println(b);
                }
                flag = true;
            }
            return;
        }
        for (int i = start; i < 9; i++) {
            if (isSelected[i]) {
                continue;
            }

            isSelected[i] = true;
            sevenGGoma[cnt] = shorts[i];
            combi(cnt + 1, i + 1);
            isSelected[i] = false;
        }
    }
}
```

발표자 : 황승연 \ 문제 : 백준 2605 줄세우기

승연님의 경우 한번 뒤집어서 입력 및 순서 변경 처리를 한 후에 다시 뒤집어서 출력함.

정필님의 경우 입력 받은 순서대로 처리해서 출력함

```
package practice;

import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
```

```
public class Boj_2605_줄세우기 {

   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);

      int N = sc.nextInt();
      int[] input = new int[N];      //입력배열
      for(int i=0;i<N;i++) {
         input[i] = sc.nextInt();
      }
      //입력 끝

      int[] arr =new int[N];         //index-input배열
      for(int i=0;i<N;i++) {
         arr[i] = i-input[i];      //index-input이 작을수록 앞쪽으로감
      }

      int num=1;
      List<Integer> list = new ArrayList<Integer>();      //줄세울 어레이리스트(순서 반대로)
      for(int i=0;i<N;i++) {
         if(arr[i]==0) {                           //0이면 맨 앞으로 가니까 리스트에 추가해서
 맨뒤로
            list.add(num++);
         }else {
            list.add(list.size()-arr[i], num++);      //0이 아니면 끝에서 index-input으로
 (반대니까)
         }
      }
      for(int i=N-1;i>=0;i--) {
         System.out.print(list.get(i)+" ");            //반대로 출력
      }
   }

}
```

발표자 오윤기 \ 문제 백준 8320 직사각형을 만드는 방법

N개의 1x1 정사각형으로 만들 수 있는 사각형의 개수는 총 몇 개인지 구하는 문제

axa인 정사각형 이전과 이후의 사각형들은 같은 사각형 이므로 axa로 표현 가능한 사각형까지만 탐색함

```
package Baekjoon;

import java.util.Scanner;

public class Main8320 {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int n = sc.nextInt();
    int cnt = 0;
```

```
    //n이하에 존재하는 모든 사각형의 개수는
    //1~n 까지의 약수와 관련이 있다
    //하지만 같은 모양의 사각형은 중복임으로
    //1부터 n까지 root n 값까지 포함되는 약수의 개수만
    //세어주면 정답이 됨.
    for (int i = 1; i <=n; i++) {
      for (double j = 1; j <=Math.sqrt(i); j++) {
        if ((i % j) == 0) {
          cnt++;
        }
      }
    }
    System.out.println(cnt);
    sc.close();
  }

}
```

발표자 : 김유진 \ 문제 : SWEA 7964 부먹왕국의차원관문

해당 문제의 경우 차원 관문이 이전 차원 관문의 범위 내에 존재한다면 최대 범위를 갱신하고

이전 차원 관문의 최대 범위까지 새로운 차원 관문이 없다면 최대 범위에 차원 관문을 건설하고 최대 범위를 갱신하는 방법을 사용했으며,

갱신되는 최대 범위가 무조건 존재하는 n+1번 위치까지 커버할 수 있는 경우에 알고리즘을 종료

```
import java.io.BufferedReader;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.StringTokenizer;

//public class Solution {
public class SWEA7964_부먹왕국의차원관문_3 {

  public static void main(String[] args) throws NumberFormatException, IOException {
    System.setIn(new FileInputStream("C:/CodingStudy/SWEA/D3/7964_input.txt"));
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    int T = Integer.parseInt(br.readLine());
    StringTokenizer st;
    for (int tc = 1; tc <= T; tc++) {
      int ans = 0;
      st = new StringTokenizer(br.readLine());
      int N = Integer.parseInt(st.nextToken());
      int D = Integer.parseInt(st.nextToken());
      int[] city = new int[N + 2]; // 0,N+1 을 고려해서 크기 결정
```

```
      city[0] = 1; // 맨처음에는 반드시 존재함
      city[N + 1] = 1; // 맨 끝 에는 반드시 존재함
      st = new StringTokenizer(br.readLine()); // 도시 정보 한줄 읽어오기
      for (int idx = 1; idx <= N; idx++) {
        city[idx] = Integer.parseInt(st.nextToken()); // 1부터 N까지 도시 정보 입력
      }
      int maxRange = 0;
      for (int i = 0; i < N + 1; i++) {

        if (maxRange == i && city[i] == 0) {// 범위 끝자락에 게이트가 없다면
          city[i] = 1; // 게이트 생성
          maxRange = i + D; // 게이트 범위 갱신
          ++ans; // 건설 횟수 증가
        }
        if (city[i] == 1) // 1이면 범위 증가
          maxRange = i + D; // 범위 갱신
        if (maxRange >= N + 1) // 범위가 끝까지인 경우
          break; // 종료

      }

      System.out.println("#" + tc + " " + ans);

    } // end tc
  }// end main

}
```