Ex1.c

```c
/* 통과 완료 */
// 익숙해지면 디버거를 사용하면 되지만, 아니라면 출력해보면 테스트
#include <stdio.h>

int main()
{
    int n;

    scanf("%d", &n);

    /* 문제 포인트 : for / while */
    /* 내 포인트는 n > 0 때까지 */
    while (n > 0) {     /* 몇 번 반복해야 되는 지 정해져 있는 경우 for 문을 쓰면 되고,
    여기서는 수학적 계산으로 for 문을 쓸 수 있겠지만... */
        n /= 2;
        printf("%d ", n);
    }

    return 0;
}
```

Ex2.c

```c
/* 통과 완료 */
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    int n;
    int num[100];  /* 배열의 길이를 변수로 지정 할 수 없음, 혹은 동적 메모리 할당,
    현재 문제에선 100 이니까, 100 에서는 메모리 소비가 적음, 상업적일 때는 신경 써봐야 함 */
    scanf("%d", &n);
    for (int i=0; i<n; i++) {
        scanf("%d", &num[i]);
    }

    /* 최솟값은 아주아주아주 기본적인 로직, 생각하지 않고 쓸 정도 */
    int theMin;
    int theSecond;
    if (num[0] <= num[1]) {
        theMin = num[0];
        theSecond = num[1];
```

```
    }
    else {
        theMin = num[1];
        theSecond = num[0];
    }
    for (int i=2; i<n; i++) {
        if (num[i] <= theMin) {     // =
            theSecond = theMin;     // 이전 최소값을 세컨값으로 둠
            theMin = num[i];
        }
        else if (num[i] < theSecond) {  // theSecond > num[i] > theMin
            theSecond = num[i];
        }
    }
    printf("%d %d\n", theMin, theSecond);
    return 0;
}
```

* 2차
모든 요소 훑는 게임
해석, 헷갈릴 필요X
요소 中 min보다 작은 값이 없다면
결국 min은 항시 sec 값이 됨
= 이 안 들어간 이유
ex) theMin = -12 였는데
num[i] = -12 였다하면
theSecond = -12가 됨
굳이 말해서 할 필요가 없다.

EX3

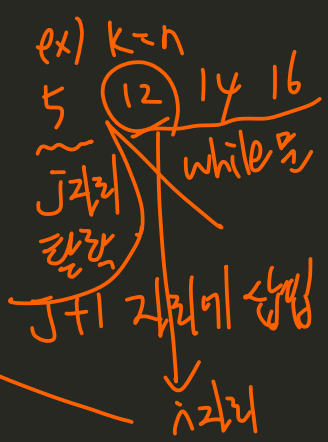
Ex3.cpp

```
/* 통과 완료 */
#include <stdio.h>

// void insert(int *n, int num[], int k);
int insert(int n, int num[], int k);

int main(void)
{
    /* 전연변수를 남발하지 마라. 는 규모가 클 때 일부분에서 그 변수를 사용할 경우
    현재 프로그램에서는 모든 부분에서 쓰이므로 오히려 더 단순화되고 사용했을 때 좋음
*/
    int num[100];
    int n = 0, k;

    /* -1 를 입력할 때 끝이 나므로 while */
    while (1) {
        scanf("%d", &k);
        if (k==-1) break;
        /* 아래는 k 를 처리하는 부분,
        main 함수에서 모든 일을 하는 것은 좋지 않음, 길어질 뿐더러 가독성 떨어짐 */
        // insert(&n, num, k);
        n = insert(n, num, k);
        // bool duplicate = false;
        // for (int i=0; i<n; i++) {
        //     if (num[i] == k) {
        //         printf("duplicate\n");
        //         duplicate = true;
```

중복체크
처음 n=0 이라서
안 돌아감

```
        //         break;
        //     }
        // }
        // /* k가 중복된 값인지 알 수 있어야함 */
        // /* 이미 정렬이 되어 있는 상태에서 추가하여 정렬 => 삽입 정렬 */
        // if (!duplicate) {
        //     int j = n-1;
        //     while (j>=0 && num[j] > k) {    /* j>=0 &&는 초보 때 금방금방 안 떠오름
        //                                        처음 조건 불만족 또는 두번째 조건 불만족으로 빠져나옴 */
        //         num[j+1] = num[j];
        //         j--;
        //     }
        //     num[j+1] = k;   //
        //     n++;    //
        //     for (int i=0; i<n; i++)
        //         printf("%d ", num[i]);
        //     printf("\n");
        // }
    }

    return 0;
}

// void insert(int *n, int num[], int k)
// {
//     for (int i=0; i<*n; i++) {
//         if (num[i] == k) {
//             printf("duplicate\n");
//             return;
//         }
//     }
//     /* 앞에서 중복이면 함수를 종료시키기 때문에 아래에서는 중복이 아닌 경우에만 실행됨 */
//     int j = *n-1;
//     while (j>=0 && num[j] > k) {
//         num[j+1] = num[j];
//         j--;
//     }
//     num[j+1] = k;   //
//     (*n)++;    //      /* 값에 의한 호출, main n의 값이 바뀌지 않음 */
//     /* 전역변수로 보내기 또는 변수의 주소를 넘겨줌 */

//     for (int i=0; i<*n; i++)
//         printf("%d ", num[i]);
//     printf("\n");
```

```c
// }

/* 매개변수 하나만 문제가 있으니까 반환값으로 내보냄 */
int insert(int n, int num[], int k)
{
    /* 어차피 배열은 오름차순으로 정렬되어 있기에 입력값을 순서대로 비교를 하다가
    입력값보다 큰 수가 나왔을 경우 그 뒤는 검사할 필요가 없음 */
    int i=0;
    for (; i<n && num[i] <= k; i++) {
        if (num[i] == k) {
            printf("duplicate\n");
            return n;
        }
    }
    /* 입력값이 들어가야 할 자리를 앎 */
    /* i 가 n 일 때 가장 큰 정수이거나  */

    // i --> position to insert

    int j = n-1;
    for (; j>=i; j--)
        num[j+1] = num[j];
    num[j+1] = k;   //
    n++;    //      /* 값에 의한 호출, main n 의 값이 바뀌지 않음 */
    /* 전역변수로 보내기 또는 변수의 주소를 넘겨줌 */

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

    return n;
}
```



Ex4.cpp

```c
/* 통과 완료 */
#include <stdio.h>

int abs(int a)
{
    if (a>=0) return a;
    return -a;
}

int main(void)
{
    int n, k;
    int num[100];
    scanf("%d", &n);
    for (int i=0; i<n; i++)
        scanf("%d", &num[i]);
    scanf("%d", &k);

    /* 합, 최소값 등등등 다 비슷한 문제, 데이터를 다 읽고 답을 갱신해나가면 되는지 */
    /* 최소차이 */

    int idx_min_diff = 0;
    for (int i=1; i<n; i++) {
        if (abs(num[i]-k) < abs(num[idx_min_diff]-k))
            idx_min_diff = i;
    }

    printf("%d", num[idx_min_diff]);

    return 0;
}
```

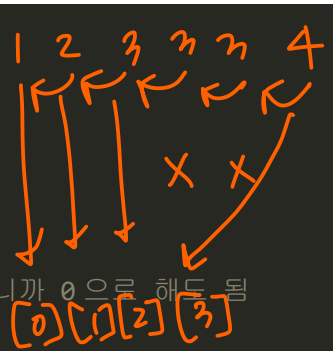
차이의 절댓값

EX5

Ex5.cpp

```c
/* 통과 완료 */
#include <stdio.h>

void bubblesort(int n, int num[]);

/* 정렬하고 중복확인하는 것이 더 효율적 */
int main(void)
{
    int n;
    int num[100];
    scanf("%d", &n);
```

```
    for (int i=0; i<n; i++)
        scanf("%d", &num[i]);

    bubblesort(n, num);

    // 1 2 3 3 3 4 4 5 5 7 8 9 9
    // 자기 바로 앞과 비교해서 삭제
    int j = 0;  // 첫번째 애는 무조건 살아남으니까 0으로 해도 됨
    for (int i=0; i<n; i++) {
        if (i==0 || num[i] > num[i-1])  // survive, 이동, 살아남은 애들의 위치를
알고 있어야 이동 가능
            num[j++] = num[i];
    }

    printf("%d: ", j);
    for (int i = 0; i < j; i++) {
        printf("%d ", num[i]);
    }

    return 0;
}

void bubblesort(int n, int num[])
{
    for (int i=n-1; i>0; i--) {
        for (int j = 0; j < i; j++) {
            if (num[j] > num[j + 1]) {
                int tmp = num[j];
                num[j] = num[j + 1];
                num[j + 1] = tmp;
            }
        }
    }
}
```



Ex6.cpp

```
/* 통과 완료 */
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

/* 프로그래머는 코딩하기 전에 문해력이 필요 */

double prob(int N, int k);

int main(void)
```

```
{
    srand((unsigned int)time(NULL));

    for (int k=1; k<=6; k++) {
        double p = prob(6 * k, k);
        printf("%lf\n", p);
    }

    return 0;
}

double prob(int N, int k)
{
    int num_sucess = 0;
    for (int e=0; e<1000000; e++) {
        int count_one = 0;
        for (int t=0; t<N; t++) {
            int r = rand()%6 + 1;
            if (r == 1)
                count_one++;
        }
        if (count_one >= k)
            num_sucess++;
    }

    return (double)num_sucess/1000000;
}
```

이함수에서는주사위를N번던져서그중1이 k번이상나오는지는검사하는실험을 1,000,000번 반복한다.

18번 던져서 1이 3번 이상 나오는 확률

Ex7

Ex7.cpp

```
/* 통과 완료 */
#include <stdio.h>
#include <string.h>

void bubblesort(int n, char *words[]);

int main(void)
{
    char *words[100];
    int n=0;
    char buf[20];

    scanf("%d", &n);

    for (int i=0; i<n; i++) {
        scanf("%s", buf);
        words[i] = strdup(buf);
```

buf는 다시 새로 입력받아야 하니까 strdup 사용

```c
    }

    bubblesort(n, words);

    for (int i = 0; i < n; i++) {
        printf("%s\n", words[i]);
    }

    return 0;
}

void bubblesort(int n, char *words[])
{
    for (int i=n-1; i>0; i--) {
        for (int j = 0; j < i; j++) {
            if (strlen(words[j]) > strlen(words[j + 1])
              || (strlen(words[j]) == strlen(words[j + 1])
              && strcmp(words[j], words[j + 1]) > 0)) {
                char *tmp = words[j];
                words[j] = words[j + 1];
                words[j + 1] = tmp;
            }
        }
    }
}
```