<12월 알고리즘 문제 풀이 답>

1.

```
#include <stdio.h>
#include <string.h>
int main() {
    int arr[10] = {0},m = -1;
    char s[101];
    scanf("%s", s);
    for(int i = 0;i<strlen(s);i++){
        arr[s[i] - '0'] += 1;
    }
    for(int i = 0;i<10;i++){
        if(arr[i] >= arr[m]){
            m = i;
        }
    }
    printf("%d", m);
    return 0;
}
```

2.

```
#include <stdio.h>
#include <stdbool.h>

int reverse(int x) {
     int num = 0;
     while (x > 0) {
          num = num * 10 + x % 10;
          x /= 10;
     }
     return num;
}

bool func(int x) {
     if (x < 2) return false;
     for (int i = 2; i * i <= x; i++)
          if (x % i == 0) return false;
     return true;
}

int main() {
     int n;
```

```
    scanf("%d", &n);

    int arr[100];
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    for (int i = 0; i < n; i++) {
        int re = reverse(arr[i]);
        if (func(re)) {
            printf("%d ", re);
        }
    }

    return 0;
```

3.

```
#include <stdio.h>

int main(){
        //freopen("input.txt", "rt", stdin);
        int i, n, m, check, count=0, max=0;
        scanf("%d %d", &n, &m);

        for(i=1; i<=n; i++){
                scanf("%d", &check);
                if(check > m){
                        count++;
                } else {
                        count = 0;
                }

                if(count > max) {
                        max = count;
                }
        }

        if(max == 0) {
                printf("-1");
        } else {
                printf("%d", max);
        }
```

```
	return 0;
}
```

4.

```
#include<stdio.h>
int main(){
	//freopen("input.txt", "rt", stdin);
	int i, j, a[200], b[200], n;
	scanf("%d", &n);
	for(i=1; i<=n; i++){
		scanf("%d", &a[i]);
		b[i]=1;
	}
	for(i=1; i<=n; i++){
		for(j=1; j<=n; j++){
			if(a[j]>a[i]) b[i]++;
		}
	}
	for(i=1; i<=n; i++){
		printf("%d ", b[i]);
	}
	return 0;
}
```

5.

```
#include <stdio.h>

int main() {
    int N, K;
    scanf("%d %d", &N, &K);

    int temperatures[N];
    for (int i = 0; i < N; i++) {
        scanf("%d", &temperatures[i]);
    }

    int max_sum = 0, current_sum = 0;

    // 초기 K일 합 계산
    for (int i = 0; i < K; i++) {
        current_sum += temperatures[i];
    }
```

```
    max_sum = current_sum;

    // 슬라이딩 윈도우로 최대 합 계산
    for (int i = K; i < N; i++) {
        current_sum += temperatures[i] - temperatures[i - K];
        if (current_sum > max_sum) {
            max_sum = current_sum;
        }
    }

    printf("%d\n", max_sum);
    return 0;
}
```

6.

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

int main() {
    int grid[9][9];

    // 9x9 격자 입력
    for (int i = 0; i < 9; i++)
        for (int j = 0; j < 9; j++)
            scanf("%d", &grid[i][j]);

    // 각 행 처리 및 출력
    for (int i = 0; i < 9; i++) {
        int sum = 0, closest = 0;

        // 합 계산
        for (int j = 0; j < 9; j++) sum += grid[i][j];

        int avg = round((double)sum / 9); // 평균 (반올림)

        // 평균과 가장 가까운 값 찾기
        for (int j = 0; j < 9; j++) {
            if (abs(grid[i][j] - avg) < abs(closest - avg) ||
                (abs(grid[i][j] - avg) == abs(closest - avg) && grid[i][j] > closest)) {
                closest = grid[i][j];
            }
```

```
        }

        printf("%d %d\n", avg, closest); // 결과 출력
    }

    return 0;
}
```

7.

```
#include <stdio.h>
#define MAX 2000
int main() {
    int N, K;
    int tasks[MAX];
    int total_time = 0;
    // 입력 받기
    scanf("%d", &N);
    for (int i = 0; i < N; i++) {
        scanf("%d", &tasks[i]);
        total_time += tasks[i]; // 작업에 필요한 총 시간 계산
    }
    scanf("%d", &K);

    // 만약 K가 총 작업 시간보다 크면 작업이 모두 끝남
    if (K >= total_time) {
        printf("-1\n");
        return 0;
    }

    int current_time = 0; // 현재 시간
    int index = 0;          // 현재 작업 번호

    // 작업 처리 시뮬레이션
    while (current_time < K) {
        if (tasks[index] > 0) { // 현재 작업이 남아 있는 경우
            tasks[index]--;     // 1초 처리
            current_time++;     // 시간 증가
        }
        index = (index + 1) % N; // 다음 작업으로 이동 (순환 구조)
    }
```

```
    // 정전 후 다시 시작할 작업 찾기
    while (tasks[index] <= 0) { // 작업이 남아 있는 경우를 찾음
        index = (index + 1) % N;
    }
    // 출력 (작업 번호는 1부터 시작하므로 +1)
    printf("%d\n", index + 1);
    return 0;
}
```