1. 진수변환과 palindrome

1.1 진수 변환

10진수 자연수 n과 k를 입력 받고 n을 k진법으로 변환한 수를 *하나의 숫자씩 공백을 넣어* 출력하는 프로그램을 작성하라.

n과 k의 범위는 다음과 같다.

$0 < n < 10^8$, $1 < k < 10$.

10진수 자연수 n이 10, k가 2일 때 2진법으로 변환한 수는 1010이므로 이를 공백을 넣어

1 0 1 0 으로

출력한다.

실행예1:

10 2 <- 입력: n과 k

1 0 1 0 <- 출력: n의 2진수

실행예2:

17 4 <- 입력: n과 k

1 0 1 <- 출력: n의 4진수

1.2 진수변환(재귀적으로 풀기)(이 문제를 풀지 않고 1.3 로 진행해도 됩니다)

1.1과 같은 일을 하는 프로그램을 재귀적 함수 transformR을 완성하여 작성하라.(빈칸 넣기)

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
int transformR(int n, int* a, int k)
{
	int result;
	if (n != 0) {
		// 빈칸 넣기: 재귀적으로 작성
	}
	return 0;
}
void testPrint(int* a, int size)
{
	for (int i = 0; i < size; i++)
		printf("%d ", a[i]);
	printf("₩n");
}
int main(void)
{
	int n, k; // 만약 10의 12승으로 하게 되면 여기 int 형을 바꾸어야…
	int a[100], size;
	scanf("%d %d", &n, &k);

	size = transformR(n, a, k);
	testPrint(a, size);

	return 0;
}
```

1.3 (Palindrome 판별)

10진수인 자연수를 k진법으로 변환한 수가 palindrome인지 알아보는 프로그램을 작성하시오. 10진수 자연수 n과 k를 입력 받고, k진법으로 변환된 수가 palindrome이면 1을 출력하고 그렇지 않으면 0을 출력한다.

$0 < n < 10^{12}$

$1 < k < 10$ or $k = 16$.

유의사항:
이 문제는 테스트케이스 중 한 개라도 틀릴 경우 0점,
즉 모두 맞춰야 만점으로 처리됩니다.

실행예1:
27 2 <- 입력: n과 k
1 <- 출력: 27을 2진수로 변환하면 1 1 0 1 1이고 1 1 0 1 1이 palindrome이므로 1을 출력한다.

실행예2:
27 3 <- 입력
0 <- 출력: 27을 3진수로 변환하면 1 0 0 0이고 1 0 0 0이 palindrome이 아니므로 0을 출력한다.

실행예3:
417 16 <- 입력
1 <- 출력: 417을 16진수로 변환하면 1 10 1이고 1 10 1이 palindrome이므로 1을 출력한다.

2. 마이너스 carry

2_1. 반복으로 풀기

항상 n1 > n2를 가정하고 n1에서 n2를 빼는 문제를 생각해보자.
두 개의 자연수를 뺄 때 한 자리씩 오른쪽에서 왼쪽으로 뺀다. 이때, 그 차가 0보다 작으면 왼쪽 자리에서 1을 꿔오는 식으로 뺄셈을 진행한다. 이처럼 1을 꿔오는 것을 minusCarry라고 하자.

두 개의 자연수 n1, n2(n1 > n2) 를 입력 받아서 뺄 때 몇 번의 minusCarry가 생기는지를 출력하는 프로그램을 작성하라.

실행예1:
729 34 <- 입력: 두 개의 자연수
1 <- 출력: minusCarry는 1개

실행예2:
100000 7 <- 입력: 두 개의 자연수
5 <- 출력: minusCarry는 5개

```
#pragma warning(disable : 6031)
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
int minusCarryCount(int n1, int n2)
{
	// 코드 작성
}
int main(void)
{
	int num1, num2;
	scanf("%d %d", &num1, &num2);
	printf("%d₩n", minusCarryCount(num1, num2));
	return 0;
}
```

2_2 함수를 재귀적으로 작성

```
#pragma warning(disable : 6031)
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int minusCarryCountR(int n1, int n2, int carry) // 이 함수를 작성하게
{
	if (n1 == 0)
		return 0;
	// 빈칸 넣기
}
int main(void)
{
	int num1, num2;
	int carry = 0;
	scanf("%d %d", &num1, &num2);
	printf("%d₩n", minusCarryCountR(num1, num2, carry));
	return 0;
}
```

3. 선택정렬의 변형

수업시간에는 다음과 같은 원형을 갖는 선택정렬을 다루었다.

void selectionSort(int A[], int n);

즉, 크기가 n인 배열 A를 선택정렬을 이용하여 정렬하였다.

이를 시작점 start를 추가하여 배열 A에서 시작점 인덱스를 start로 하여 배열의 마지막까지 정렬하는 (부분 배열 A[start … n-1]을 정렬하는) 함수 selectionSortNew로 변경하라.

크기 n이 5인 배열 a가 40 30 20 10 50로 입력된 경우
start를 1로 입력 받으면
배열 인덱스 1부터 마지막까지 정렬하면
40 10 20 30 50이 출력된다.

실행 예1:
5 <- 입력: 배열의 개수 n
40 30 20 10 50 <- 입력: 배열 a의 원소 값
1 <- 입력: 정렬을 시작하는 인덱스 start
40 10 20 30 50 <- 출력: 인덱스 1부터 정렬

실행 예2:
5 <- 입력: 배열의 개수 n
40 30 20 10 50 <- 입력: 배열 a의 원소 값
2 <- 입력: 정렬을 시작하는 인덱스 start
40 30 10 20 50 <- 출력: 인덱스 2부터 정렬

```
void testPrint(int* a, int size)
{
	for (int i = 0; i < size; i++)
		printf("%d ", a[i]);
	printf("₩n");
}
void selectionSortNew(int A[], int left, int n)
{
	// 코드 작성
}
int main(void)
{
	int n;
	int a[100];
	int start;
	scanf("%d", &n);
	for (int i = 0; i < n; i++)
		scanf("%d", &a[i]);
	scanf("%d", &start);

	selectionSortNew(a, start, n);
	testPrint(a, n);
	return 0;

}
```

4. (등차수열)

4.1 등차수열 판별

주어진 배열이 등차 수열인가를 판별하는 함수 isSequence를 작성하라. 이 함수의 원형은 다음과 같다.
int isSequence(int a[], int size);

배열의 개수 n가 5이고
입력되어진 배열 a의 값이 10 12 14 16 18이면
등차 수열이므로 1을 출력한다.

배열의 개수가 n이 5일 때
입력되어진 배열 a의 값이 10 12 13 15 17이면
등차수열이 아니므로 0을 출력한다.

유의사항:
이 문제는 테스트케이스 중 한 개라도 틀릴 경우 0점,
즉 모두 맞춰야 만점으로 처리됩니다.

실행 예1:
5 <- 입력: n
10 12 14 16 18 <- 입력: 배열 a의 값
1 <- 출력: 등차수열이다

실행 예2:
5 <- 입력: n
10 12 13 16 18 <- 입력: 배열 a의 값
0 <- 출력: 등차수열이 아니다.

((초기 코드로 준다))

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
int isSequence(int* a, int size)
{
	// 코드 작성
}

int main(void)
{
	int n;
	int* a;

	scanf("%d", &n);
	a = (int*)malloc(sizeof(int) * n);

	for (int i = 0; i < n; i++)
		scanf("%d", &a[i]);
	printf("%d₩n", isSequence(a, n));
	free(a);
	return 0;
}
```

4.2 (뽑기)등차수열 개수

크기가 n인 임의의 숫자들로 구성되어 있는 배열의 값을 조금씩(1, 혹은 -1) 변경하려 한다. 입력 배열의 각각의 수에 연산을 최대 한 번 수행할 수 있다. 연산의 종류는 1을 더하거나 1을 빼는 것이 있다.

입력 배열의 개수와 배열의 각각의 수를 입력 받아

위와 같은 연산을 통해서 입력배열이 등차수열이 되는 경우의 수를 출력하라.

크기가 4인 배열 10 12 13 16의 경우 위에서 제시한 연산을 적용하면

다음과 같은 2개의 등차수열이 가능하므로 경우의 수 2가 출력된다.

9 11 13 15(-1 -1 0 -1을 적용)

10 12 14 16(0 0 +1 0을 적용)

실행 예1:

4 <- 입력: n

10 12 13 16 <- 입력: n개의 배열 원소 값

2 <- 출력: 위의 연산 수행 후 등차수열이 되는 경우는 2가지

실행 예1:

4 <- 입력: n

10 12 50 60 <- 입력: n개의 배열 원소 값

0 <- 출력: 위의 연산 수행 후 등차수열이 되는 경우는 없음

int main(void) // 이 main 함수를 그대로 사용해도 좋고, 변경하여도 좋다

```
{
        int items[3] = { -1, 0, 1 };
        int n;
        int* bucket;
        int* num1;
        int* num2;

        //printf("개수 입력");
        scanf("%d", &n);

        bucket = (int*)malloc(sizeof(int) * n);
        num1 = (int*)malloc(sizeof(int) * n);
        num2 = (int*)malloc(sizeof(int) * n);

        for (int i = 0; i < n; i++)
                scanf("%d", &num1[i]);
        printf("%d₩n", pick(items, 3, bucket, n, n, num1, num2));
}
```

4.3(뽑기)등차수열 중 최소 연산 개수
위의 문제에서처럼 입력 배열에 대해 연산을 수행할 때,
등차수열로 변환하기 위한 필요한 최소 연산 횟수를 출력하라.
입력 배열의 개수와 배열의 각각의 수를 입력하면 최소 연산 횟수를 출력한다.
등차수열로 만들 수 없으면 -1을 출력한다.

앞의 문제에서 보았듯이
크기가 4인 배열 10 12 13 16의 경우 위에서 제시한 연산을 적용하면
다음과 같은 2개의 등차수열이 가능하다.
9 11 13 15(-1 -1 0 -1을 적용한 3개의 연산)
10 12 14 16(0 0 +1 0을 적용한 1개의 연산)
그러므로 최소 연산의 개수로 1을 출력한다.

실행예1:
4 <- 입력: n
10 12 13 16 <- 입력: n개의 배열 원소 값
1 <- 출력: 등차배열을 만드는 최소 연산의 개수

실행예2:
4 <- 입력: n
24 21 15 9 <- 입력: n개의 배열 원소 값
3 <- 출력: 배열에 연산(+1 -1 0 +1)을 적용하면 25 20 15 10인데 이 경우가 유일하므로 연산의 (최소)개수는 3이다.

실행예3:
4 <- 입력: n
99 3 2 1 <- 입력: n개의 배열 원소 값
-1 <- 출력: 등차수열로 만들 수 없는 입력 배열이다.

3.2(다음 순열 나열)
(위에서 작성한 변형된 선택정렬을 사용해도 좋고, 물론 안 해도 좋다)
1부터 N까지의 수로 이루어진 순열이 있다. 순열 중의 한 배열을 입력으로 받아 사전순으로 다음에 오는 배열을 출력하시오. 사전 순으로 가장 앞서는 순열은 오름차순으로 이루어진 순열이고, 가장 마지막에 오는 순열은 내림차순으로 이루어진 순열이다.

N = 3인 경우에 사전 순으로 순열을 나열하면 다음과 같다.
1, 2, 3 → 1, 3, 2 → 2, 1, 3 → 2, 3, 1 → 3, 1, 2 → 3, 2, 1

N = 4인 경우는 다음과 같다.
1 2 3 4 → 1 2 4 3 → 1 3 2 4 → 1 3 4 2 → 1 4 2 3 → 1 4 3 2 … → 4 3 2 1

입력은 N과 순열 중의 임의의 배열이며 출력은 그 다음 배열이다. 출력은 1줄에 해야 하고 숫자와 숫자 사이에는 공백이 한 칸만 있어야 한다. 다음 배열이 존재하지 않으면 -1을 출력한다.

힌트*((이걸 주면 어떨지 싶습니다. 사실 저도 이 규칙을 찾는데 조금 시간이 걸려서요. ^^ 시험 때는 더 쫓길듯해서))*
이 문제를 푸는 방법은 여러 가지가 있을 수 있겠으나
한가지 방법을 제시하면 다음과 같습니다. (물론 학생들은 자유로이 문제를 풀어도 됩니다.)
이 문제를 푸는 방법은 다음의 3가지로 이루어진다.

1. 일년의 숫자들의 뒷부분에서 내림차순의 최대 sequence를 찾아 그 바로 앞 인덱스 pivot으로 설정한다. 즉, 배열 A[]가 10 20 50 40 30인 경우
   최대 시퀀스는 50 40 30이 되고 그 바로 앞 인덱스는 1(20의 값을 가진)가 된다.
2. 인덱스 pivot의 뒷부분에서 A[pivot]보다 크면서 가작 작은 수를 (A[pivot] 다음으로 큰 수, A[]에서는 30) 찾아 A[pivot]과 자리바꿈을 한다.
3. 인덱스 (pivot + 1) 지점부터 뒷부분까지 (A[pivot+1 … n-1]을) 정렬한다.

실행예1:
3 <- 3개의 숫자
1 3 2 <- 입력 배열
2 1 3 <- 1 3 2의 다음은 2 1 3

실행예2:
3 <- 3개의 숫자
3 2 1 <- 입력 배열
-1 <- 3 2 1 다음은 없으므로

실행예3:
4 <- 3개의 숫자
1 3 4 2 <- 입력 배열
    1 4 2 3 <- 답

3.3 3번 문제를 1부터 N까지의 N개 숫자에서 M개의 숫자를 조합의 형태로 뽑아서 사전 순으로 나열한다고 했을 때 입력한 배열 다음으로 나오는 배열을 출력하시오. (M <= N)