입력한 int형 정숫값이 음수이면 -1을, 0이면 0을, 양수이면 1을 반환하는 signOf 메서드를 작성하자.

실행 예1

```
정수 x : 13
singOf(x)는 1입니다.
```

실행 예2

```
정수 x : 0
singOf(x)는 0입니다.
```

실행 예3

```
정수 x : -15
singOf(x)는 -1입니다.
```

3개의 int형 인수 a, b, c 중 최솟값을 구하는 min 메서드를 작성하자.

```
int min(int a, int b, int c)
```

실행 예

```
정수 a: 1
정수 b: 3
정수 c: 2
최솟값은 1입니다.
```

3개의 int형 인수 a, b, c에서 중간값을 구하는 med 메서드를 작성하자.

```
int med(int a, int b, int c)
```

실행 예

```
정수 a: 1
정수 b: 3
정수 c: 2
중간값은 2입니다.
```

1부터 n까지의 정수의 합을 구해서 반환하는 메서드를 작성하자.

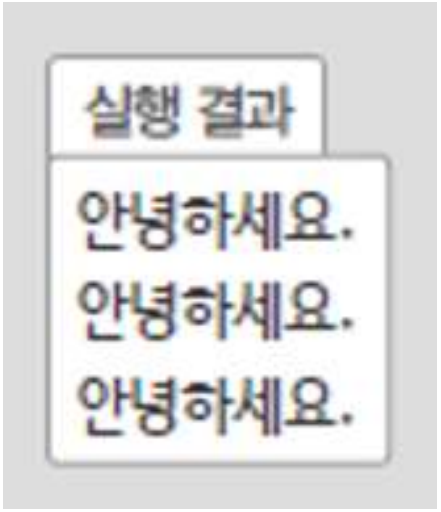
```
int sumUp(int n)
```

실행 예

```
1부터 x까지의 합을 구하자.
x의 값: 5
1부터 5까지의 합은 15입니다.
```

"안녕하세요."라고 표시하는 hello 메서드를 작성하자.

```
void hello()
```



인수 m에 지정한 달(月)의 계절을 표시하는 printSeason 메서드를 작성하자. m값에 따라 봄(3, 4, 5), 여름(6, 7, 8), 가을(9, 10, 11), 겨울(12, 1, 2) 표시하고 그 이외의 값이 오면 아무것도 표시하지 않는다.

```
void printSeason(int m)
```

실행 예

```
몇 월입니까(1 ~ 12) : 4
해당 월의 계절은 봄입니다.
```

문자 c를 n개 표시하는 putChar 메서드와 이 메서드를 내부에서 호출해서 문자 '*'를 n개 연속으로 표시하는 putStart 메서드를 작성하자. 또한, 이 메서드들을 사용해서 직삼강형을 만드는 프로그램을 작성하자.

실행 예

```
왼쪽 아래가 직각인 이등변 삼각현을 표시합니다.
단수는:6
*
**
***
****
*****
******
```

난수(a =< 난수 =<b)를 생성해서 반환하는 random 메서드를 작성하자. 메서드 안에서 난수를 생성하는 표준 라이브러리를 호출할 것. (참고, b <= a인 경우 a값을 그대로 반환할 것)

```
int random(int a, int b)
```

실행 예

```
난수를 생성합니다.
하한값:10
상한값:99
생성한 난수는 52입니다.
```

"양의 정숫값 : "이라는 메세지에 정숫값을 입력하면 값을 거꾸로 반화하는 readPlusInt를 메서드를 작성하자. 0이나 음수(-)가 입력되면 재입력하도록 할 것.

```
int readPlusInt()
```

실행 예

```
양의 정숫값 :-125
양의 정숫값 :0
양의 정숫값 :521
반대로 읽으면 125입니다.
다시 한 번?<Yes_1/No_0>:
```

다음 4개의 계산 문제 중 하나를 무작위로 출제하는 암산 훈련 프로그램을 작성하자. 이때 x, y, z에 3자리의 정숫값을 난수로 생성할 것.

x + y + z　　x + y – z　　x – y + z　　x – y – z

실행 예

```
암산 훈련!!
341 + 616 - 742 = 215
틀렸습니다!
341 + 616 - 742 = 216
다시 한 번?<Yes…1/No…0>:1
674 + 977 + 760 = 2411
다시 한 번?<Yes…1/No…0>:0
```

정수를 좌우로 시프트한 값이, '정수 X 2의 거듭제곱' 및 '정수 / 2의 거듭 제곱'과 같은지 확인하는 프로그램을 작성하자.

실행 예

```
정수x를 n비트 시프트합니다.
x:100
n:3
[a] x × (2의 3승) = 800
[b] x ÷ (2의 3승) = 12
[c] x ≪ 3 = 800
[d] x ≫ 3 = 12
[a]와 [c]의 값이 일치합니다.
[b]와 [d]의 값이 일치합니다.
```

정수 x를 오른쪽으로 n비트 회전한 값을 반환하는 rRotate 메서드와 왼쪽으로 n비트 회전한 값을 반환하는 lRotate 메서드를 작성하자.

```
int rRotate(int x, int n)        int lRotate(int x, int n)
```

· 회전이란 최하위 비트와 최상위 비트가 연결돼 있다고 간주하는 것이다. 예를 들어 오른쪽으로 5비트 회전한 경우, 시프트에 방출된 하위 5비트를 상위 비트로 다시 가져온다.

실행 예

```
정수x를 n비트 회전합니다.
x:1565857138
n:6
회전 전 =
01011101010101010001010101110010
오른쪽 회전 =
11001001011101010101010001010101
왼쪽 회전 =
01010101010001010101110010010111
```

정수 x의 pos번째 있는 비트(최하위 비트부터 0, 1, 2,…)를 1로 변경한 값을 반환하는 set 메서드, 0으로 변경한 값을 반환하는 reset 메서드, 그리고 해당 위치의 비트의 반전시켜서 반환하는 inverse를 작성하자.

```
int set(int x, int pos)        int reset(int x, int pos)        int inverse(int x, int pos)
```

실행 예

```
정수x의 pos번째 비트를 변경합니다.
x:1431655765
pos:10
x                =    01010101010101010101010101010101
set(x, pos)      =    01010101010101010101010101010101
reset(x, pos)    =    01010101010101010100010101010101
inverse(x, pos) =    01010101010101010100010101010101
```

정수 x의 pos번째에 있는 비트부터 n개 연속되는 비트를 1로 변경한 값을 반환하는 setN 메서드, 0으로 변경한 값을 반환하는 resetN, 반전시킨 값을 반환하는 inverseN을 작성하자.

```
int set(    int x, int pos, int n)        int reset(int x, int pos, int n)
int inverse(int x, int pos, int n)
```

실행 예

```
정수x의 pos번째 비트부터 n개 비트를 변경합니다.
x:1431655765
pos:7
n:6
x                 = 01010101010101010101010101010101
setN(x, pos, n)   = 01010101010101010101111111010101
resetN(x, pos, n) = 01010101010101010100000001010101
inverse(x, pos, n) = 01010101010101010100101011010101
```

배열 a이 가진 모든 요소의 합을 구하는 sumOf() 메서드를 작성하자.

```
int sumOf(int[] a)
```

실행 예

```
요소 수:5
x[0]:22
x[1]:5
x[2]:11
x[3]:32
x[4]:120
모든 요소의 합은 190입니다.
```

배열 a의 요소 중에서 최솟값을 구하는 minOf 메서드를 작성하자.

```
int minOf(int[] a)
```

실행 예

```
사람 수는:4
4명의 신장과 체중을 입력하자.
1번의 신장:175
1번의 체중:72
2번의 신장:163
2번의 체중:82
3번의 신장:150
3번의 체중:49
4번의 신장:181
4번의 체중:76
가장 키가 작은 사람의 신장:150cm
가장 마른 사람의 체중:49kg
```

배열 a로부터 key와 같은 값을 가지는 요소를 탐색하는 linearSearch 메서드와 linearSearchR 메서드를 작성하자. 단, 키와 같은 값을 가지는 요소가 여러 개인 경우 linearSearch는 가장 앞에 위치한 요소를 찾으며, linearSeachR은 가장 뒤에 위치한 요소를 찾을 것.

```
int linearSearch(int[] a, int key)
int linearSearchR(int[] a, int key)
```

실행 예

```
요소 수:6
x[0]:5
x[1]:22
x[2]:74
x[3]:32
x[4]:120
x[5]:22
찾는 값:22
해당 값의 요소가 여러 개 존재합니다.
가장 앞에 위치한 값은 x[1]에 있습니다.
가장 뒤에 위치한 값은 x[5]에 있습니다.
```

선형 탐색

배열 a로부터 요소 a[idx]를 삭제하는 aryRmv 메서드를 작성하자.

```
void aryRmv(int[] a, int idx)
```

a[idx]의 삭제는, 그 뒤에 있는 요소들을 앞으로 하나씩 이동해서 할 것. 이동한 후에 비게 되는 마지막 요소 a[a.length - 1]의 값은 이동하기 전의 마지막 값을 유지할 것.

예) 배열 a의 요소가 {1, 3, 4, 7, 9, 11}일 때에 aryRmv(a, 2)라고 호출한 후에는 배열 a의 요소는 {1, 3, 7, 9, 11, 11}이 된다.

실행 예

```
요소 수:6
a[0]:1
a[1]:3
a[2]:4
a[3]:7
a[4]:9
a[5]:11
삭제할 요소의 인덱스:2
a[0] = 1
a[1] = 3
a[2] = 7
a[3] = 9
a[4] = 11
a[5] = 11
```

배열 a에서 요소 a[idx]부터 n개의 요소를 삭제하는 aryRmvN 메서드를 작성하자.

```
void aryRmv(int[] a, int idx, int n)
```

삭제는 a[idx]보다 뒤에 있는 모든 요소를 하나씩 앞으로 이동해서 할 것. 이동 대상이 아닌 요소는 기존 값을 그대로 유지할 것.

예) 배열 a의 요소가 {1, 3, 4, 7, 9, 11}일 때 aryRmvN(a, 1, 3)라고 호출하면 배열 a의 요소는 {1, 9, 11, 7, 9, 11}이 된다.

실행 예

```
요소 수:6
a[0]:1
a[1]:3
a[2]:4
a[3]:7
a[4]:9
a[5]:11
삭제를 시작할 인덱스:1
삭제할 요소의 수:3
a[0] = 1
a[1] = 9
a[2] = 11
a[3] = 7
a[4] = 9
a[5] = 11
```

배열 a의 요소 a[idx]에 x를 삽입하는 aryIns 메서드를 작성하자.

```
void aryIns(int[] a, int idx, int x)
```

삽입 시에는 a[idx] ~ a[a.length-2]를 하나씩 뒤로 이동시켜야 한다.

예) 배열 a의 요소가 {1, 3, 4, 7, 9, 11}일 때에 aryIns(a, 2, 99)라고 호출하면 a의 요소가 {1, 3, 99, 4, 7, 9}가 된다.

실행 예

```
요소 수:6
a[0]:1
a[1]:3
a[2]:4
a[3]:7
a[4]:9
a[5]:11
삽입할 요소의 인덱스:2
삽입할 값:99
a[0] = 1
a[1] = 3
a[2] = 99
a[3] = 4
a[4] = 7
a[5] = 9
```

배열 a와 배열 b의 전체 요솟값을 교환하는 aryExchng 메서드를 작성하자.

```
void aryExchng(int[] a, int[] b)
```

두 배열의 요소 수가 같지 않다면 작은 쪽의 배열 수에 맞추어 교환할 것.

예) 배열 a의 요소가 {1, 2, 3, 4, 5, 6, 7}이고 배열 b의 요소가 {5, 4, 3, 2, 1}일 때에, aryExchange(a, b)를 호출하면 배열 a는 {5, 4, 3, 2, 1, 6, 7}이 되고 배열 b는 {1, 2, 3, 4, 5}가 돼야 한다.

실행 예

```
배열 a의 요소 수:7
a[0]:1
a[1]:2
a[2]:3
a[3]:4
a[4]:5
a[5]:6
a[6]:7
배열 b의 요소 수:5
b[0]:5
b[1]:4
b[2]:3
b[3]:2
b[4]:1
배열 a와 b의 전체 요소를 교환했습니다.
a[0] = 5
a[1] = 4
a[2] = 3
a[3] = 2
a[4] = 1
a[5] = 6
a[6] = 7
b[0] = 1
b[1] = 2
b[2] = 3
b[3] = 4
b[4] = 5
```

배열 a와 같은 배열(요소 수가 같고 모든 요소의 값이 같은 배열)을 생성해서 반환하는 arrayClone 메서드를 작성하자.

```
int[] arrayClone(int[] a)
```

실행 예

```
요소 수:5
x[0]:11
x[1]:22
x[2]:33
x[3]:44
x[4]:55
배열 x를 복사해서 배
열 y를 생성했습니다.
y[0] = 11
y[1] = 22
y[2] = 33
y[3] = 44
y[4] = 55
```

배열 a의 요소 중에서 값이 x인 모든 요소의 인덱스를 앞에서부터 순서대로 저장해서 반환하는 arraySrchIdx를 작성하자.

```
int[] arraySrchIdx(int[] a)
```

예) 배열 a의 요소가 {1, 5, 4, 8, 5, 5, 7}이고 arraySrchIdx(a, 5)를 호출한 경우, 반환할 배열은 {1, 4, 5}가 된다(값이 5인 요소의 인덱스를 나열한 것).

실행 예

```
요소 수:7
x[0]:1
x[1]:5
x[2]:4
x[3]:8
x[4]:5
x[5]:5
x[6]:7
탐색할 값:5
일치하는 요소의 인덱스
1
4
5
```

배열 a에서 요소 a[idx]를 삭제한 배열을 반환하는 arrayRmvOf를 작성하자.

```
int[] arrayRmvOf(int[] a, int idx)
```

삭제는 a[idx]보다 뒤에 있는 모든 요소를 하나씩 앞으로 이동시킬 것.

예) 배열 a의 요소가 {1, 3, 4, 7, 9, 11}일 때에 arrayRmvOf(a, 2)를 호출한 경우, 반환할 배열의 요소는 {1, 3, 7, 9, 11}이다.

실행 예

```
요소 수:6
x[0]:1
x[1]:3
x[2]:4
x[3]:7
x[4]:9
x[5]:11
삭제할 요소의 인덱스:2
y[0] = 1
y[1] = 3
y[2] = 7
y[3] = 9
y[4] = 11
```

배열 a에서 요소 a[idx]부터 n개의 요소를 삭제한 배열을 반환하는 arrayRmvOfN 메서드를 작성하자.

```
int[] arrayRmvOfN(int[] a, int idx, int n)
```

삭제는 a[idx]보다 뒤에 있는 모든 요소를 n개 앞으로 이동해서 할 것.

예) 배열 a의 요소가 {1, 3, 4, 7, 9, 11}일 때에 arrayRmvOfN(a, 1, 3)을 호출한 경우, 반환할 배열의 요소는 {1, 9, 11}이다.

실행 예

```
요소 수:6
x[0]:1
x[1]:3
x[2]:4
x[3]:7
x[4]:9
x[5]:11
삭제를 시작할 인덱스:1
삭제할 요소의 개수:3
y[0] = 1
y[1] = 9
y[2] = 11
```

배열 a의 요소 a[idx]에 x를 삽입해서 배열로 반환하는 arrayInsOf를 작성하자.

```
int[] arrayInsOf(int[] a, int idx, int x)
```

삽입할 때는 a[idx]뒤에 있는 모든 요소를 하나씩 뒤로 이동시킬 것.

예) 배열 a의 요소가 {1, 3, 4, 7, 9, 11}일 때에 arrayInsOf(a, 2, 99)를 호출할 경우, 반환할 배열의 요소는 {1, 3, 99, 4, 7, 9, 11}이 된다.

실행 예

```
요소 수:6
x[0]:1
x[1]:3
x[2]:4
x[3]:7
x[4]:9
x[5]:11
삽입할 인덱스:2
삽입할 값:99
y[0] = 1
y[1] = 3
y[2] = 99
y[3] = 4
y[4] = 7
y[5] = 9
y[6] = 11
```

행렬 x와 y의 합을 구해서 z에 저장하는 addMatrix 메서드를 작성하자.

```
boolean addMatrix(int[][] x, int[][] y, int[][] z)
```

3개 배열의 요소 수가 같으면 계산해서 true를 반환하고 같지 않으면 계산없이 false를 반환할 것.

실행 예

```
행렬a
1 2 3
4 5 6

행렬b
6 3 4
5 1 2

행렬c
7 5 7
9 6 8
```

행렬 x와 y의 합을 저장해서 2차원 배열로 반환하는 메서드를 작성하자(행 수 및 열 수가 동일한 x, y를 전달한다고 가정해도 좋다).

```
int[][] addMatrix(int[][] x, int[][] y)
```

실행 예

```
행렬의 행 수:2
행렬의 열 수:3
a[0][0]:1
a[0][1]:2
a[0][2]:3
a[1][0]:4
a[1][1]:5
a[1][2]:6
b[0][0]:6
b[0][1]:3
b[0][2]:4
b[1][0]:5
b[1][1]:1
b[1][2]:2
행렬a
1 2 3
4 5 6

행렬b
6 3 4
5 1 2

행렬c
7 5 7
9 6 8
```

2차원 배열 a와 동일한 배열(요소 수가 같고 모든 요소의 값도 같다)을 생성해서 반환하는 arryClone2를 작성하자.

```
int[][] aryClone2(int[][] a)
```

실행 예

```
행렬의 행 수:2
행렬의 열 수:3
a[0][0]:1
a[0][1]:2
a[0][2]:3
a[1][0]:4
a[1][1]:5
a[1][2]:6
행렬a
1 2 3
4 5 6
행렬a의 복사본
1 2 3
4 5 6
```

2개의 int형 정수 a, b의 최솟값, 3개의 int형 정수 a, b, c의 최솟값, 배열 a의 최솟값을 각각 구하자. 다음에 정의된 메서드 형식을 사용할 것.

`int min(int a, int b)` `int min(int a, int b, int c)` `int min(int[] a)`

실행 예

```
x값:13
y값:52
z값:11
배열 a의 요소 수:4
a[0]:3
a[1]:7
a[2]:1
a[3]:8
x, y의 최솟값은 13입니다.
x, y, z의 최솟값은 11입니다.
배열 a의 최솟값은 1입니다.
```

int형 변수 x의 절댓값, long형 변수 x의 절댓값, float형 변수 x의 절댓값, double형 변수 x의 절댓값을 구하는 다중 메서드를 작성하자.

```
int absolute(int x)          long absolute(long x)
float absolute(float x)      double absolute(double x)
```

실행 예

```
int    형 정수 a의 값:5
long   형 정수 b의 값:-8
float  형 실수 c의 값:-13.5
double 형 실수 d의 값:27.4
a의 절댓값은 5입니다.
b의 절댓값은 8입니다.
c의 절댓값은 13.5입니다.
d의 절댓값은 27.4입니다.
```

int형의 1차원 배열과 int형의 2차원 배열(행에 따라 열 수가 다를 가능성이 있음)의 모든 요솟값을 표시하는 다중 정의 메서드를 작성하자.

`void printArray(int[] a) void printArray(int[][] a)`

1차원 배열 표시에선 각 요소 사이에 문자 1개분의 공간을 둘 것. 또한, 2차원 배열 표시에선 각 열의 숫자가 왼쪽에 맞추어 정렬되도록 최소한의 공간을 둘 것.

예를 들면 다음과 같이 표시

1차원 배열의 표시 예

```
12 536 -8 7 2
```

2차원 배열의 표시 예

```
32  -1    32   45 67
535 99999 2
2   5     -123 9
```

실행 예

```
 1차원 행렬 x의 요소 수:4
x[0]:12
x[1]:536
x[2]:-8
x[3]:7
 2차원 배열 y의 행 수:3
0행째 열 수:5
1행째 열 수:3
2행째 열 수:4
각 요소의 값을 입력하자.
y[0][0]:32
y[0][1]:-1
y[0][2]:32
y[0][3]:45
y[0][4]:67
y[1][0]:535
y[1][1]:99999
y[1][2]:2
y[2][0]:2
y[2][1]:5
y[2][2]:-123
y[2][3]:9

 1차원 배열x
12 536 -8 7

 2차원 배열y
32  -1    32   45 67
535 99999 2
2   5     -123 9
```