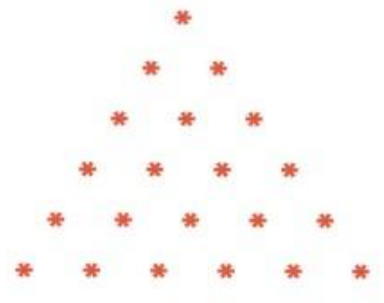
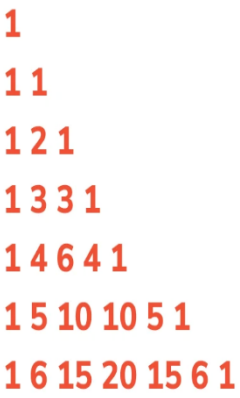
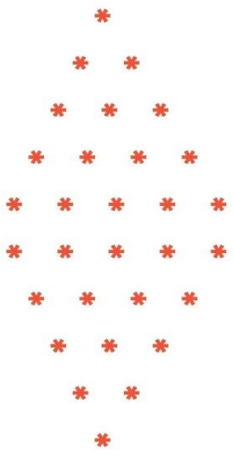
Pattern printing problems:-

1. Pyramid Pattern Programs in C or C++



Full Pyramid





Solid Diamond

2. Diagonal & Sides of a Rectangle

time complexity analysis:-

1. What is the time, and space complexity of the following code:

```
int a = 0, b = 0;
for (i = 0; i < N; i++) {
    a = a + rand();
}
for (j = 0; j < M; j++) {
    b = b + rand();
}
```

Options:

1. O(N * M) time, O(1) space
2. O(N + M) time, O(N + M) space
3. O(N + M) time, O(1) space
4. O(N * M) time, O(N + M) space

2.What is the time complexity of the following code:

```
int i, j, k = 0;
for (i = n / 2; i <= n; i++) {
    for (j = 2; j <= n; j = j * 2) {
        k = k + n / 2;
    }
}
```

Options:

1. O(n)
2. O(N log N)
3. O(n^2)
4. O(n^2Logn)

3. Analyze and find the time complexity of the below function from the code snippet.

```
int k=1;
while(k<=n){
        cout<<k<<endl;
        k=k*2;
}
```

4. Analyze and find the time complexity of the below function from the code snippet.

```
for(int i=n/2;i<=n;i++){
        for(int j=1;j<=n;j=j*2){
                cout<<i<<j<<endl;
        }
}
```

Linear search:-

1. Given a List of Distinct N number a1,a2,a3.....an.

Find The Position Of Number K In The Given List.

Input Format

First Line Take Input Value Of N

Second Line Take Input N Space Separated Integer Value

Third Line Take Input Value Of K

Constraints

0 < N < 100001

0 < ai < 10^10

0 < K < 10^10

Output Format

Position Of K In The Given List.

Print -1 if element is not found.

Sample Input 0

```
5
1 2 3 4 5
4
```

Sample Output 0

```
3
```

Circular array loop questions:-

You are playing a game involving a **circular** array of non-zero integers nums. Each nums[i] denotes the number of indices forward/backward you must move if you are located at index i:

- If nums[i] is positive, move nums[i] steps **forward**, and
- If nums[i] is negative, move nums[i] steps **backward**.

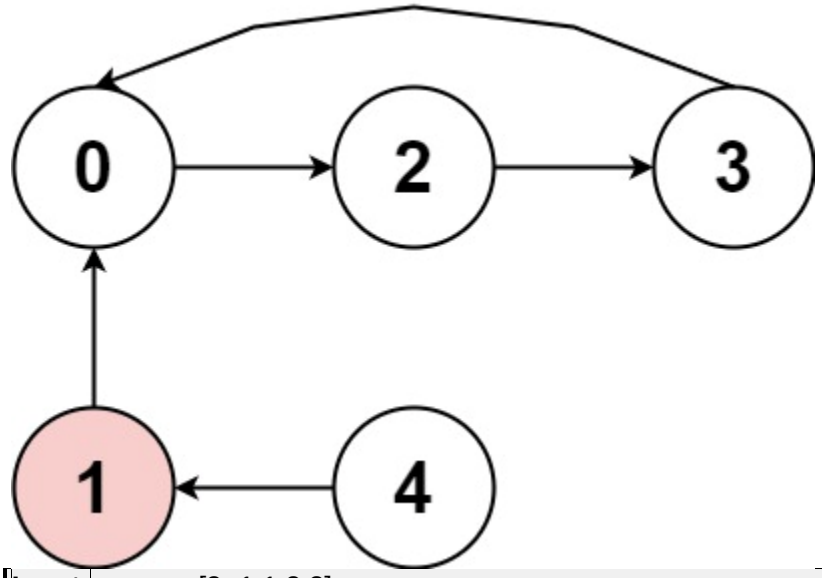
Since the array is **circular**, you may assume that moving forward from the last element puts you on the first element, and moving backwards from the first element puts you on the last element.

A **cycle** in the array consists of a sequence of indices seq of length k where:

- Following the movement rules above results in the repeating index sequence seq[0] -> seq[1] -> ... -> seq[k - 1] -> seq[0] -> ...
- Every nums[seq[j]] is either **all positive** or **all negative**.
- k > 1

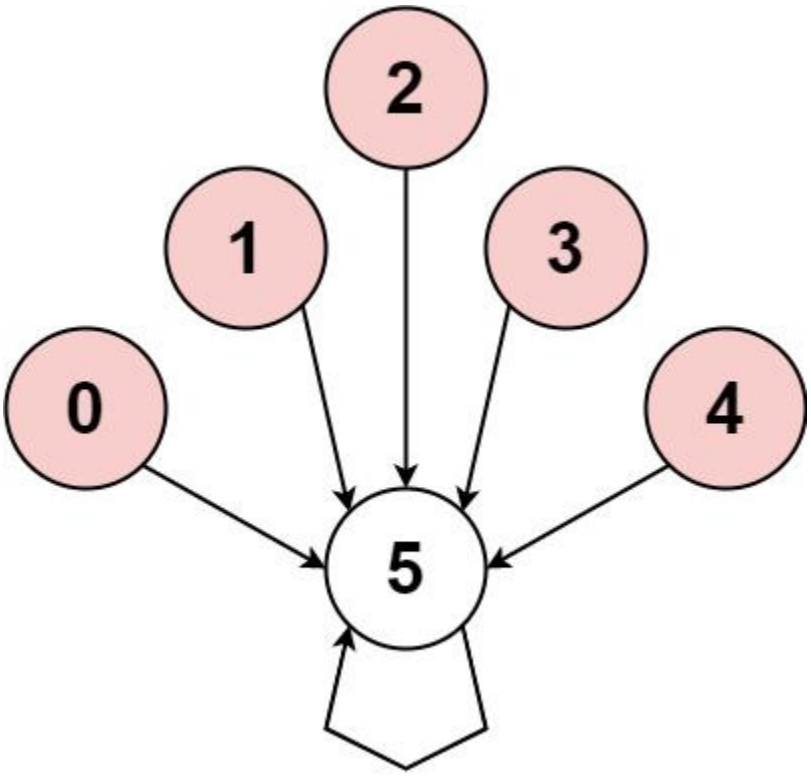
Return true *if there is a **cycle** in* nums*, or* false *otherwise*.

Example 1:



Input: nums = [2,-1,1,2,2]
Output: true

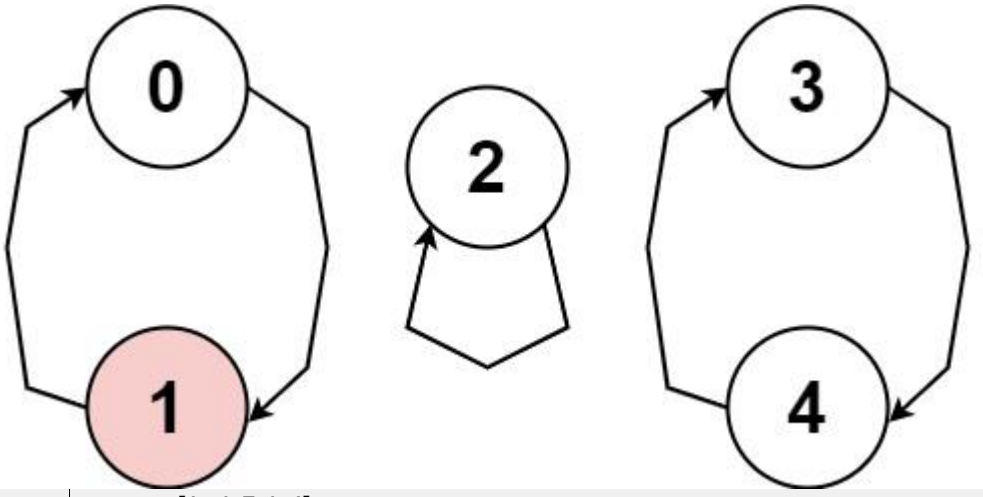
Example 2:



Input: nums = [-1,-2,-3,-4,-5,6]

Output: false

Example 3:



Input: nums = [1,-1,5,1,4]
Output: true

Constraints:

- 1 <= nums.length <= 5000
- -1000 <= nums[i] <= 1000
- nums[i] != 0