Question - 1
Caesar Cipher

Your task is to implement the earliest and simplest cipher called **Caesar Cipher**, which each letter in a text is shifted by a certain number of places down/up in the alphabet.
For example,
with a shift of 1, A would be replaced by B, B would become C, and so on.

Write a program which performs the Caesar encryption for a text. Your program should read the input from the user from STDIN, perform the cipher operation and finally output the encoded text to STDOUT.
You are **only allowed to use <stdio.h> library** for your implementation.

The first line of input contains the number of shifts.
From second line onwards input contains the text (Note that text can have any number of input lines).
The output format is given in the shown examples when you run the caesar program.

For example,

```
./caesar
1
HAI
Here is the encoded text:
IBJ

./caesar
-5
Hello!
How are you?
Here is the encoded text:
Czggj!
Cjr vmz tjp?
```

You must implement the following three functions to achieve this task.

- *int rotateright (int ch)*
  This function rotates a specified alphabetic character one step to the right, and returns the new character.
  Non-alphabetic characters should be returned without change.

- *int rotateleft(int ch)*
  This function performs the inverse of rotate_right().

- *int encode(int ch, int shift)*
  This function performs the Caesar encryption for a single character by rotating the specified number of times (shift).
  The function should repeatedly call ***rotateright()*** or ***rotateleft()*** depending on whether the shift is positive or negative. (If the shift is zero, the original character should be returned.)

Help

Question - 2

Count Occurrences Extended

This is an extension to the problem **Count Occurrences** completed in Lab 04.

You are required to solve the same problem by including a function that takes a word and a text as inputs and returns the number of occurrences of the given word in the given text with the function description as below.

int occur (char word[ ], char text[ ]){}

Furthermore, you must solve the same problem without using any library functions that you used in your previous implementation. That is, the **only library** that you can use for your implementation is **<stdio.h>**. Every other function that you used for your previous implementation must be written by you this time.

For example,
Inbuilt functions from **<string.h>** library : **strlen(), strcmp(), strtok(), strcpy() and etc** should be written in your own as separate functions in your program. You can refer the function descriptions for implementation, but you need to implement the functions as your own work.

The problem description of Count Occurrences is shown below again for reference.

Count Occurrences

Write a program to find the occurrences of a given word in a given text. Your program should take inputs from STDIN and output the answer to STDOUT. Your program should take two inputs; word and text. Then, produce the output in the format below.

If any of the inputs are missing, you should display a message as "Wrong Input".

```
I
I hear and I forget, I see and I remember, I do and I understand.
The occurrences of "I" in the text are 6.
```

Question - 3

Maximum Difference in an Array

The **maximum difference** for a pair of elements in some array a is defined as the **largest difference between any a[i] and a[j]** where **i < j and a[i] < a[j].**

Write a program that calculates and returns the maximum difference for a; if no such number exists (e.g.: if a is in descending order and all a[j] < a[i]), return -1. You should implement **at least two functions** within your program and you are free to decide their descriptions your own.

Input Format
Reading input from STDIN. The first line contains N (the number of elements in the array a). The N subsequent lines each contain a single element of a; the ith line of input (where 0 < i < N-1) contains element a[i].

Output Format
Printing to STDOUT maximum difference in a.

Sample Input 0
7

2
3
10
2
4
8
1

Sample Output 0
8

Sample Input 1
6
7
9
5
6
3
2

Sample Output 1
2

Explanation
Sample Case 0: n = 7, a = {2, 3, 10, 2, 4, 8, 1}
As a[2] = 10 is largest element in the array, we must find the smallest a[i] where $0 \leq i < 2$. This ends up being 2 at index 0.
We then calculate the difference between the two elements: a[2] – a[0] = 10 – 2 = 8, and return the result (8).

Note: While the largest difference between any two numbers in this array is 9 (between a[2] = 10 and a[6] = 1), this cannot be our maximum difference because the element having the smaller value (a[6]) must be of a lower index than the element having the higher value (a[2]).
As 2 is not less than 6, these elements cannot be used to calculate our maximum difference.

Sample Case 1: n = 6, a = {7, 9, 5, 6, 3, 2}
The maximum difference returned by our function is a[1] – a[0] = 9 – 7 = 2, because 2 is the largest difference between any a[i] and a[j] satisfying the conditions that a[i] < a[j] and i < j.