**Task 2**

Language: JavaScript. Time complexity: O(N^2) Space complexity: O(N)

**Code:**

//create a javascript arrow function to generate pattern

const iLikePattern = (N,T)=>{

let pattern = ''

let isLatter = isNaN(T)

//check pattern type number or char

let startIndex = isLatter ? T.charCodeAt(0) : parseInt(T) ;

//print first line

for(let i=0;i<N;i++){

pattern += isLatter ? String.fromCharCode(startIndex +i) : startIndex +i;

}

console.log(pattern)

//print middile lines with spaces

for(let j=1;j<=N-2;j++){

pattern = ''

for(let i=0; i<N;i++){

if(i==0){

pattern += isLatter ? String.fromCharCode(startIndex +i+j) : startIndex +i+j;

}else if(i==N-1){

pattern += isLatter ? String.fromCharCode(startIndex +i-j) : startIndex + i-j;

}else{

pattern += ' '

}

}

console.log(pattern)

}

//print last line with reversed order.

pattern =''

for(let i=N-1;i>=0;i--){

pattern += isLatter ? String.fromCharCode(startIndex +i) : startIndex +i;

}

console.log(pattern)

}

// user input

let N = prompt("Enter Number: ");

let T = prompt("Enter Pattern Type(Number or char): ");

iLikePattern(Number(N),T)

Output:



