```c
// C function to reverse an input array
#include <stdio.h>
void func(int *array,int size);
int main() {
   int size,i;
   scanf("%d",&size);
   int arr[size];
   for(i=0;i<size;i++){
      scanf("%d",&arr[i]);
   }
   func(arr,size);
    return 0;
}
void func(int *array,int size){
    int i ,temp;
    for(i=0;i<size/2;i++){
      temp = array[i];
      array[i]=array[size-1-i];
      array[size-1-i]=temp;
   }
   for(i=0;i<size;i++){
      printf("%d",array[i]);
   }
}
```

```
/tmp/O7ln2lg4JI.o
4
1
2
3
4
4321

=== Code Execution Successful ===
```

Write a C function that reverse an input array

Example:

Input: 1,2,3,4,5,6

output: 6,5,4,3,2,1

Given an integer, *num*, we want to know the value of the 4th least significant bit in *num*'s binary representation. For example, if *num* = $(23)_{10}$ we first convert it to its binary representation, $(10111)_2$. When we count the bits from least to most significant, we see that the 4^{th} least significant bit is *0*.

Complete the function in the editor below. It has the following parameter:

Name	Type	Description
`num`	`integer`	The number we want the 4^{th} least significant bit for.

The function must return a binary integer (i.e.: *0* or *1*) denoting the 4^{th} least-significant bit of *num*.

Input Format

A single integer denoting *num*.

Constraints

num is a *32*-bit integer.

```c
#include <stdio.h>
int fourthBit(int i);
int main() {
    int i;
    scanf("%d",&i);
    printf("\n%d",fourthBit(i));
}
int fourthBit(int i){
    if((i & 8) == 0){
        return 0;
    }else{
        return 1;
    }
}
```

☆ Check if a given number is a power of 3

Write a C function that return 0 if a given number is a power of 3, otherwise return 1 (except 3 to the power 0)

Example:

9 ==> 0

20 ==> 1

```c
#include <stdio.h>
#include <math.h>
int power3(int i);
int main() {
    int i;
    scanf("%d",&i);
    printf("\n%d",power3(i));
}
int power3(int i){
    float x;
    x = log(i)/log(3);
    if(x == (int)x){
        return 0;
    }else{
        return 1;
    }
}
```

Write a C function to return the index of LAST occurrence of a number in a given array (index starting from 0, i.e. C array style), if the item is not in the list return -1

Example:

Array = {1,2,3,4,5,6,4}, the number is 4 ==> result = 6

```c
#include <stdio.h>
#include <math.h>
int Last(int arr[],int size,int number);
int main() {
    int size=7,num=4;
    int arr[]={1,2,3,4,5,6,4};
    printf("\n%d",Last(arr,size,num));
}
int Last(int arr[],int size,int number){
    int i,flag = 0;
    for(i=size-1;i>=0;i--){
        if(arr[i] == number){
            flag = 1;
            return i;
            break;
        }
    }
    if(flag == 0){
            return -1;
        }
    }

```

☆ Clear a specified bit in a given number

Write a C function that clears a specified bit in a given number (bit number starts from 0), if not possible return the same number as is.

Example:
Input Number = 3
Bit position = 0
==>
result = 2

```c
#include <stdio.h>
#include <math.h>
int clearBit(int num,int bit);
int main() {
	int num,bit;
	scanf("%d",&num);
	scanf("%d",&bit);
	printf("%d",clearBit(num,bit));
}
int clearBit(int num,int bit){
	return num &=~(1<<bit);
	}

```

Q1)Write a C program takes string from the user and check if it the same USERNAME or not.

```c
#include <stdio.h>
#include <string.h>
int compare(char str[],char username[]);
int main() {
    char str1[] ="kirolos";
    char str2[] = "kirolos";
    printf("%d",compare(str1,str2));
}
int compare(char str[],char username[]){
    int flag = 0,i;
    if(strlen(str)==strlen(username)){
        for(i=0;i<strlen(str)-1;i++){
            if(str[i]!=username[i]){
                flag =1;
                break;
            }
        }
        return flag;
    }
    }

```