Experiment No-1

Experiment name: Write a C program that reads a year from the keyboard and determine whether it is a leap year or not.

Objective:

- To write a C program that reads a year and determines whether it is a leap year or not.

Problem analysis:

Input Variable	Processing Variables	Output Variable	Header Files
year (integer)	Check if year is divisible by 4, 100, and 400	Display whether the year is leap year or not	#include <stdio.h>

Algorithm:

Step-1: Start

Step-2: Read year from user

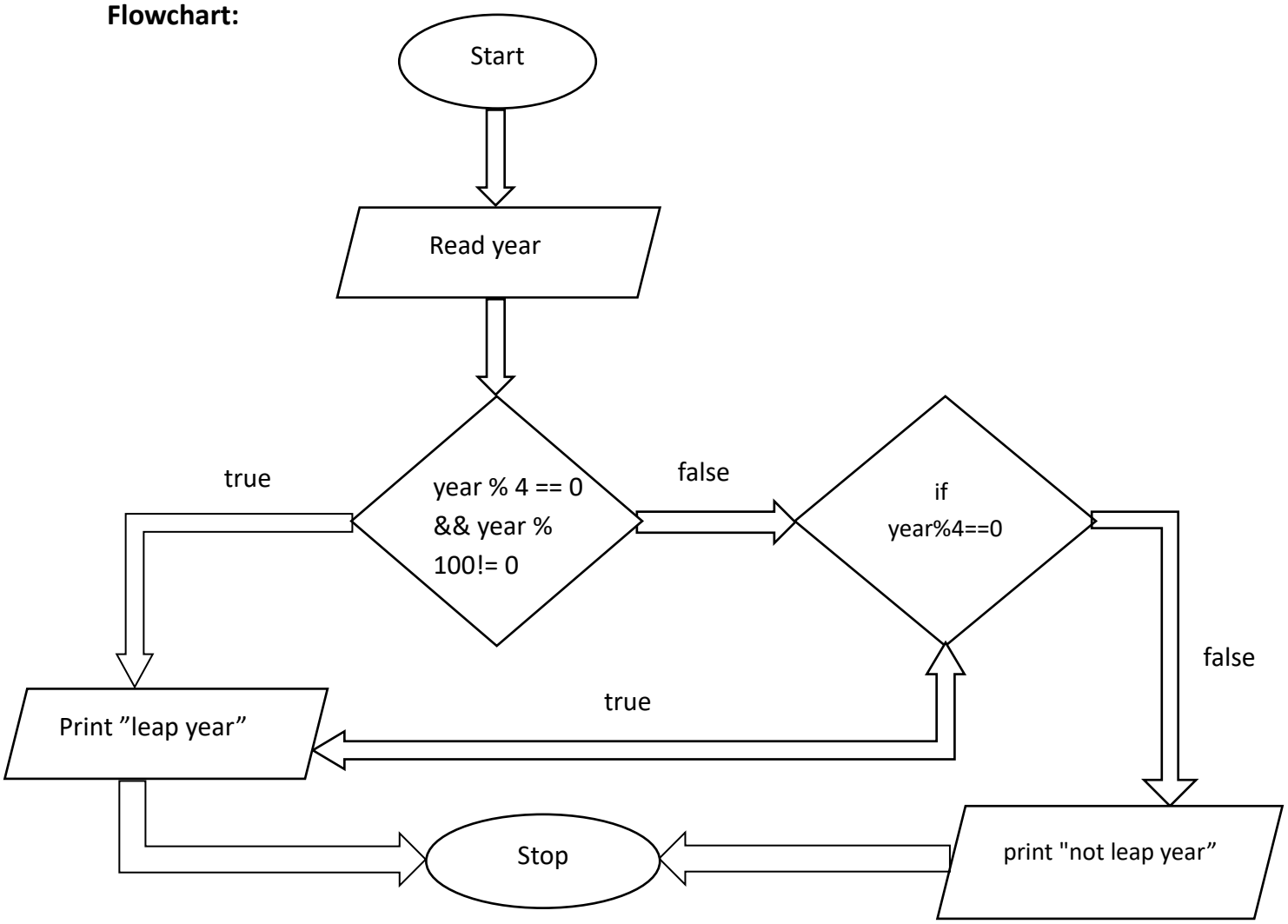
Step-3: If year is divisible by 400 → leap year

- Else if divisible by 100 → not leap year
- Else if divisible by 4 → leap year
- Else → not leap year

Step- 4: Print result

Step-5: Stop

Flowchart:



Source Code:

```c
#include <stdio.h>

int main() {
    int year1, year2;

    printf("Enter first year: ");
    scanf("%d", &year1);

    printf("Enter second year: ");
    scanf("%d", &year2);

    if ((year1 % 4 == 0 && year1 % 100 != 0) || (year1 % 400 == 0)) {
        printf("%d is a leap year.\n", year1);
    } else {
        printf("%d is not a leap year.\n", year1);
    }
    if ((year2 % 4 == 0 && year2 % 100 != 0) || (year2 % 400 == 0)) {
        printf("%d is a leap year.\n", year2);
    } else {
        printf("%d is not a leap year.\n", year2);
    }

    return 0;
}
```

Output:

```
Select "D:\c pro reatke\bin\Debug\c pro reatke.exe"
Enter first year: 2024
Enter second year: 2026
2024 is a leap year.
2026 is not a leap year.

Process returned 0 (0x0)   execution time : 8.791 s
Press any key to continue.
```

Discussion:

The program checks whether a given year is a leap year or not using the conditions of divisibility by 4, 100, and 400. It applies if-else statements and modulus operations to display the correct result, demonstrating the use of decision-making in C programming.

Experiment No-2

Experiment Name: Write a C Program to check whether a given number is a prime number or not.

Objective:

- To write a C program to check whether a given positive integer is a prime number.

Problem analysis:

In this program, the main objective is to check whether a given number is a prime number or not. The user inputs an integer value which is stored in the variable num. To determine if it is prime, the program checks how many numbers divide it exactly (remainder zero). A loop runs from 1 to num, and each time the number divides exactly, a counter variable count is increased by one. After the loop ends, if the value of count is equal to 2, it means the number is divisible only by 1 and itself, so it is a prime number. Otherwise, it is not a prime number. The program uses the header file <stdio.h> for input and output functions like printf() and scanf().

Input Variable	Processing Variables	Output Variable	Header Files
num (integer, > 0)	i (loop counter), isPrime (flag: 1 for prime, 0 otherwise)	"n is prime" or "n is not prime"	#include <stdio.h>, #include <math.h>

Algorithms:

Step-1: Start the program.

Step-2: Declare integer variables: num

Step-3: Read the value of num from the user.

Step-4: Use a loop (for) to check divisibility from i = 1 to i <= num.

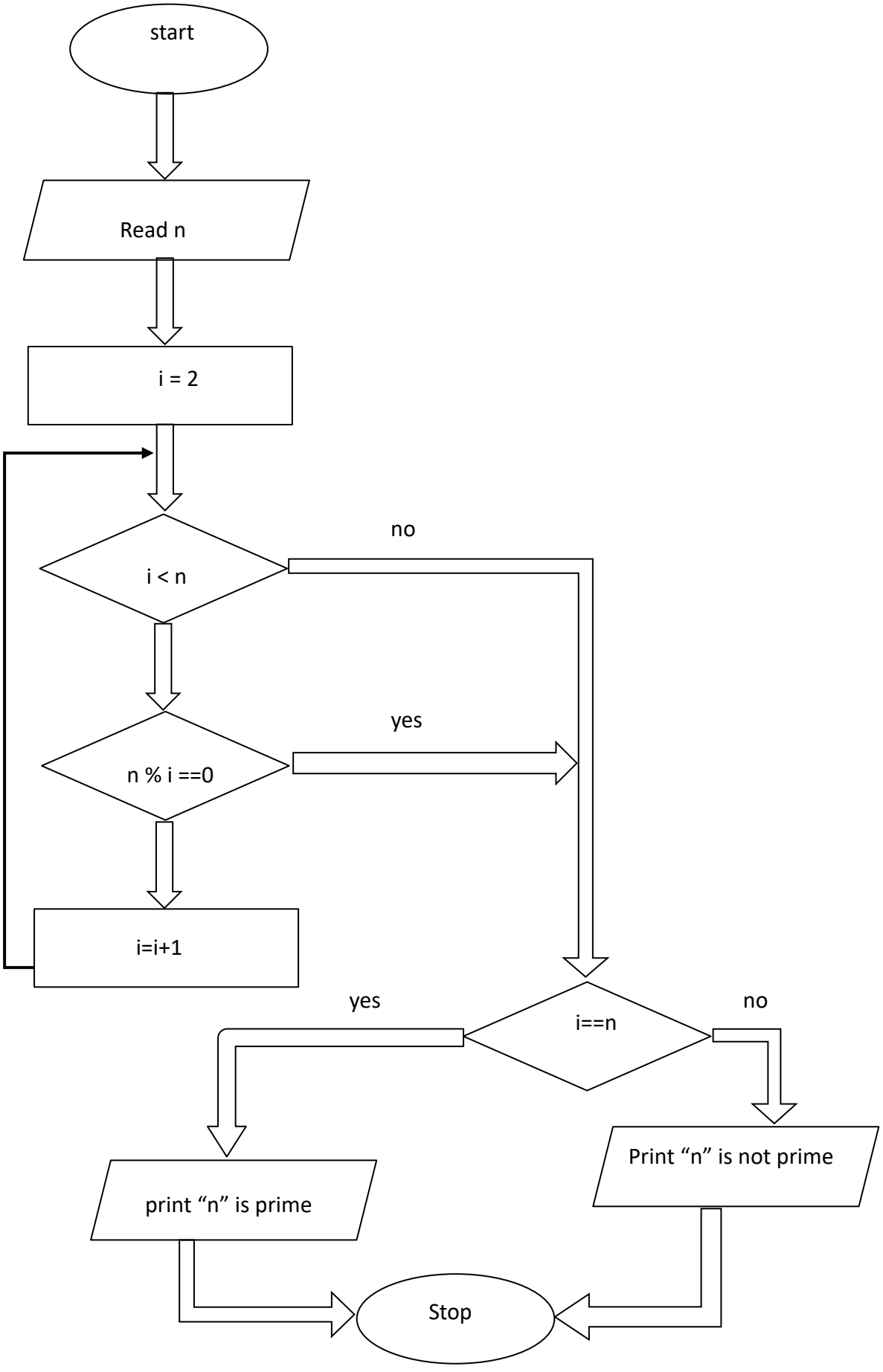
Step-5: If num % i == 0, increment count.

Step-6: After the loop ends,

- If count == 2, then print "Prime number".
- Else print "Not a prime number".

Step-7: Stop the program.

Flowchart:


start
Read n
i = 2
i < n
no
n % i ==0
yes
i=i+1
i==n
yes
no
print “n” is prime
Print “n” is not prime
Stop

Source code:

```c
#include <stdio.h>
#include <math.h>
int main() {
    int n, i, count;
    printf("How many numbers you want to check? ");
    scanf("%d", &count);
    for (int k = 1; k <= count; k++) {
        printf("Enter a positive integer: ");
        scanf("%d", &n);
        int isPrime = 1;
        if (n <= 1) {
            isPrime = 0;
        } else if (n == 2) {
            isPrime = 1;
        } else if (n % 2 == 0) {
            isPrime = 0;
        } else {
            for (i = 3; i * i <= n; i += 2) {
                if (n % i == 0) {
                    isPrime = 0;
                    break;
                }
            }
        }
        if (isPrime)
            printf("%d is a prime number.\n", n);
        else
            printf("%d is not a prime number.\n", n);
    }
    return 0; }
```

Output:

```
"D:\c pro reatke\bin\Debug\c pro reatke.exe"
How many numbers you want to check? 4
Enter a positive integer: 1
1 is not a prime number.
Enter a positive integer: 2
2 is a prime number.
Enter a positive integer: 17
17 is a prime number.
Enter a positive integer: 25
25 is not a prime number.

Process returned 0 (0x0)   execution time : 18.201 s
Press any key to continue.
```

Discussion:

In this experiment, we checked whether a number is prime by testing if it has any divisor other than 1 and itself. A prime number has exactly two factors, so the program first handles special cases like numbers ≤ 1, which are not prime, and 2, which is the only even prime number. For numbers greater than 2, the program checks divisibility from 3 up to the square root of the number using a loop. If any divisor is found, the number is not prime. This method makes the program efficient because checking up to √n reduces unnecessary calculations. Overall, the experiment demonstrates the use of conditional statements, loops, and basic mathematical logic in C programming.