

K L UNIVERSITY
FRESHMAN ENGINEERING DEPARTMENT
A Project Based Lab Report
On
DIVISIBILITY RULES

SUBMITTED BY:

ID NUMBER

2200031257

STUDENT NAME

ABHISHEK KUMAR

UNDER THE ESTEEMED GUIDANCE OF

Mr. E Rajesh Kumar

Assistant Professor.

Dept of Basic Engineering & Sciences.



KL UNIVERSITY

Green fields, Vaddeswaram – 522 502

Guntur Dt. AP, India.

DEPARTMENT OF BASIC ENGINEERING SCIENCES 1



CERTIFICATE

This is to certify that the project-based laboratory report entitled “DIVISIBILITY RULES” submitted by ,

ABHISHEK KUMAR [2200031257], to the Department of Basic Engineering Sciences, KL University in partial fulfillment of the requirements for the completion of a project-based Laboratory in “Computational Thinking for Structured Design” course in the I B Tech I Semester, is a bonafide record of the work carried out by her under my supervision during the academic year 2021 – 2022.

PROJECT SUPERVISOR

Mr. E Rajesh Kumar

HEAD OF THE DEPARTMENT

Dr. D. HARITHA

ACKNOWLEDGEMENTS

It is great pleasure for me to express my gratitude to our honorable President **Sri. Koneru Satyanarayana**, for giving me the opportunity and platform with facilities in accomplishing the project-based laboratory report.

I express my sincere gratitude to our Director **Dr. A. Jagadeesh** for his administration of our academic growth.

I express sincere gratitude to our Coordinator and HOD-BES **Dr. D. Haritha** for her leadership and constant motivation in successful our academic semester. I record it as my privilege to deeply thank you for providing us with the efficient faculty and facilities to make our ideas into reality.

I express my sincere thanks to our project supervisor **Mr. E Rajesh Kumar** for her novel association of ideas, encouragement, appreciation, and intellectual zeal which motivated us to venture into this project successfully.

Finally, it is pleased to acknowledge the indebtedness to all those who devoted themselves directly or indirectly to making this project report successful.

Name:

ABHISHEK KUMAR [2200031257]

ABSTRACT

A divisibility rule is a shorthand way of determining whether a given number is divisible by a fixed divisor, usually by performing simple calculations. There is no similarity among these rules, in the sense that for example, the rule for testing the divisibility by 3 is very different from the rule for 7. In this working paper, this working paper presents a general rule for testing divisibility by two-digitizing

The general rule is characterized by a pair of conditions, but in some cases, just the main condition of the two is necessary and sufficient for divisibility. In the second part, we investigate this aspect and show in general the cases in which just one condition is enough. Although there are divisibility tests for numbers in any base, they are usually different.

INDEX

S.NO	TITLE	PAGE NO
1	Introduction	6-7
2	Aim of the Project	8
2.1	Advantages & Disadvantages	8
2.2	Future Implementation	8
3	Software & Hardware Details	8
4	Data Flow Diagram	9
5	Algorithm for each module	10-11
6	Implementation	12-16
7	Integration and System Testing	17
8	Conclusion	18

INTRODUCTION

In this question we have used the following concepts:

LOOPS AND FUNCTIONS

LOOPS: A Loop executes the sequence of statements many times until the stated condition becomes false. A loop consists of two parts, a body of a loop and a control statement. The control statement is a combination of some conditions that direct the body of the loop to execute until the specified condition becomes false. The purpose of the loop is to repeat the same code several times.

There are different kinds of loops and functions in the C language

1. While loop:

A while loop in C programming repeatedly executes a target statement as long as a given condition is true.

Syntax of while loop:

```
while(condition) {  
    Statement;  
    increment;  
}
```

In our program, we predominantly used for loop.

2. Functions:

Functions are of 4 types. They are: -

a) with argument and with return value

b) with argument and without return value

c) without argument and with return value

d) without argument and return value

In our program we used arguments and return values which means we need to declare a function first then; we need to use it. And with arguments and with return value means we need to give input values and we can expect the return value.

For the convert function, we used with argument and without return values. We take the arguments i.e.

inputs and we do not return any value.

Function With Arguments and Return value:

Syntax of functions:

Function declaration: `int function (int);`

Function Call: `function (x);`

Function Definition:

```
int function (x) {  
    statements;  
    return x;  
}
```

RULES OF DIVISIBILITY:

Divisibility by 2: A number is divisible by 2 if and only if its last digit is divisible by 2 or in other words, is even.

Divisibility by 3: A number is divisible by 3 if and only if the sum of its digits is divisible by 3.

Divisibility by 4: A number is divisible by 4 if and only if its last two digits form a number that is divisible by 4.

Divisibility by 5: A number is divisible by 5 if and only if its last digit equals 5 or 0.

Divisibility by 6: A number is divisible by 6 if and only if it is divisible by 2 and 3 simultaneously (that is, if the last digit is even and the sum of all digits is divisible by 3).

Divisibility by 7: Vasya doesn't know such a divisibility rule.

Divisibility by 8: A number is divisible by 8 if and only if its last three digits form a number that is divisible by 8.

Divisibility by 9: A number is divisible by 9 if and only if the sum of its digits is divisible by 9.

Divisibility by 10: A number is divisible by 10 if and only if its last digit is a zero.

Divisibility by 11: A number is divisible by 11 if and only if the sum of digits on its odd positions either equals the sum of digits on the even positions, or they differ in a number that is divisible by 11.

AIM

Advantages: -

- * TO PREPARE A BARCODE

- * A divisibility rule is a shorthand way of discovering whether a given number is divisible by a fixed divisor without performing the division, usually by examining its digits.

Disadvantages: -

→ The major disadvantage of these divisibility rules is that if a number is given in a decimal system, we need to first express the number in a different base.

→ Expressing it in base $n-1$ or $n+1$ may turn out to be more expensive.

Future enhancements: -

When a number is given, we should be able to determine easily whether the given number is divisible by the given divisor or not.

SYSTEM REQUIREMENTS

➤ SOFTWARE REQUIREMENTS:

The major software requirements of the project are as follows:

Language: Turbo-C

Operating system: Windows XP or later.

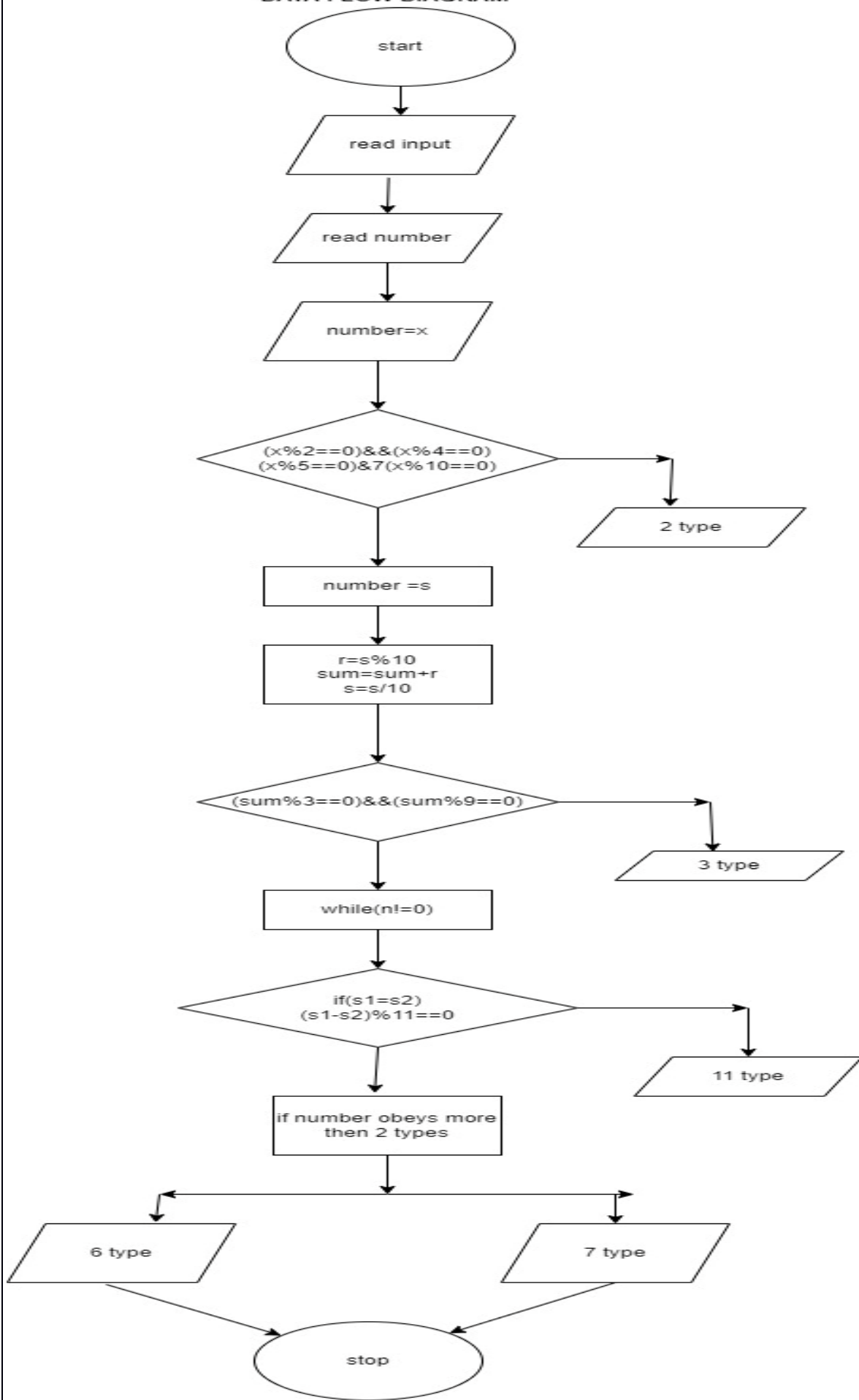
➤ HARDWARE REQUIREMENTS:

The hardware requirements that map to the software are as follows:

RAM: 8GB

Processor: intel core 13/i5

DATA FLOW DIAGRAM



ALGORITHM

STEP 1: Start

STEP 2: Declare variables that are used in the program

STEP 3: Enter the number along with the base for which you want to check the Divisibility type.

STEP 4: if base != 10 convert (b, d)

STEP 5:

5.1: s=number%10; //last digit of number

s%2==0; //divisible by 2

5.2: s=number%100; //last two digits of number

s%4==0; //divisible by 4

5.3: s=number%1000; //last three digits of number

s%8==0; //divisible by 8

5.4: s=number%10; //last digit of number

s==0 || s==5; //divisible by 5

5.5: s=number%10; //last digit of number

S==0; //divisible by 10

STEP 6: If any of the above conditions satisfy Print "2 types";

STEP 7: else

7.1: Take the number and assign it to any variable

for example: x = number;

7.2: while (x != 0) // Loop will continue until x value is 0.

7.3: r= x%10; // r-remainder

7.4: sum = sum + r; // sum will store the digits which are added

7.5: x=x/10; //we get an integer that is the next number to add

7.6: `sum%3==0 || sum%9==0; //divisible by 3 and 9`

STEP 8: If the above statements are satisfied Print “3 types “

STEP 9: else

9.1: To check the divisibility of 11

Declare counter =1; //for the given number we should count from units' places called as 1st position.

9.2: while (n!=0) // loop runs until number become zero.

9.3: if (counter%2==0) { // position is even

Even sum+=n%10;

n/=10; //even place digits sum takes place}

9.4: else {

Odd sum+=n%10;

n/=10; // odd place digits sum takes place}

9.5: counter ++ //meanwhile to change position

9.6: if(counter%2==0) {

Temp =Odd sum;

Odd sum=Even sum;

Even sum=temp;}

9.7: Even Sum - Odd sum =t; // difference value

9.8: if(t%11==0) Print “11 type”

STEP 10: If more than 2 types the number obeys then

Print “6 types”

STEP 11: If no type of rule is obeyed then

Print “7 types”

STEP 12: Stop

IMPLEMENTATION

```
#include<stdio.h>

int x;

void convert(int s,int b)
{
    x=0;
    if(s==0)
    {
        x=x*10+s;
    }
    else
    {
        convert(s/b,b);
        x=x*10+s%b;
    }
}

int a2type(int n);
int b3type(int n);
int c11type(int n);

int main()
{
    int b,d,temp=0,n,r;

    printf("Enter base and divisor\n");
```

```

scanf("%d%d",&b,&d);
if(b!=10)
{
    convert(d,b);
}
else
{
    x=d;
}
n=x;
temp+=a2type(x);
temp+=b3type(x);
temp+=c11type(x);
if(temp==2)
{
    while(n!=0)
    {
        r=n%10;
        n=n/10;
    }
    printf("2-type\n%d",r);
}
if(temp==3)
{

```

```
    printf("3-type");
}
if(temp==11)
{
    printf("11-type");
}
if(temp==5||temp==16)
{
    printf("6-type");
}
if(temp==0)
{
    printf("7-type");
}
}
int a2type(int n)
{
    int n1;
    n1=n%10;
    if(n1%2==0 || n1==5 || n1==0)
    {
        return 2;
    }
    n1=n%100;
```



```
    if(n1%4==0)
    {
        return 2;
    }
    n1=n1%1000;
    if(n1%8==0)
    {
        return 2;
    }
    return 0;
}
int b3type(int n)
{
    int s1=0;
    while(n!=0)
    {
        s1+=n%10;
        n=n/10;
    }
    if(s1%3==0||s1%9==0)
    {
        return 3;
    }
    return 0;
}
```

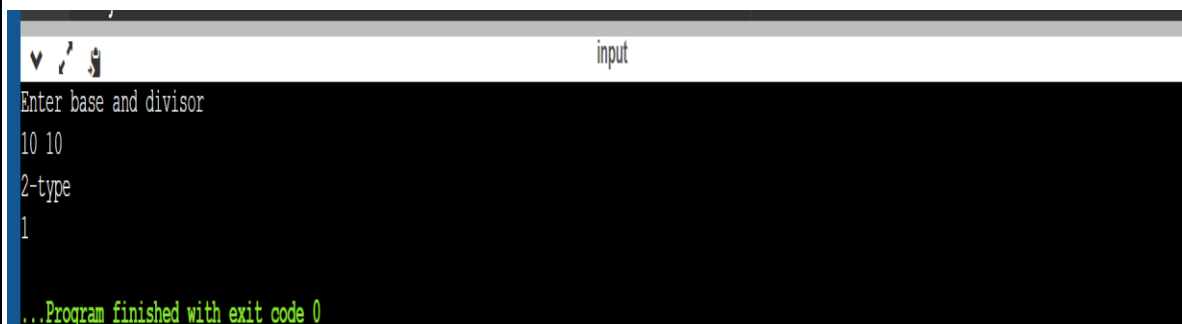
```
}  
  
int c11type(int n)  
{  
    int s1=0,s2=0;  
    while(n!=0)  
    {  
        s1+=n%10;  
        n=n/10;  
        s2+=n%10;  
        n=n/10;  
    }  
    if(s1==s2||((s1-s2)%11)==0)  
    {  
        return 11;  
    }  
    return 0;  
}
```

INTEGRATION AND SYSTEM TESTING

OUTPUTS

Screen Shots:

Sample output 1:



```
input
Enter base and divisor
10 10
2-type
1
...Program finished with exit code 0
```

Sample output 2:



```
input
Enter base and divisor
2 3
11-type
...Program finished with exit code 0
Press ENTER to exit console.
```

CONCLUSION

We completed our project “DIVISIBILITY RULES” in the allotted time. We completed our project with teamwork, we divided our project into parts & completed it successfully without any errors under the supervision of Mrs. Yamini mam.