# [04 - Iteration Control Structures](https://www.rajalakshmicolleges.net/moodle/course/view.php?id=84#section-4)

# Ex. No. : 4.1 Date:

# Register No.: Name:

# In mathematics, the factorial of a non-negative integer n, denoted by n!, is the product of all

# positive integers less than or equal to n. For example,

# 5! = 5 x 4 x 3 x 2 x 1 = 120

# 4! = 4 x 3 x 2 x 1 = 24

# 9! = 9 x 8 x 7 × 6 × 5 x 4 x 3 x 2 x 1 = 362880

# Write a program to find the factorial of a given number.

# The given number will be passed to the program as an input of type int.

# The program is expected to calculate the factorial of the given number and return it as an

# int type.

# Assumptions for this program:

# The given input number will always be greater than or equal to 1.

# Due to the range supported by int. the input numbers will range from 1 to 12.

# For example:

|  |  |
| --- | --- |
| Input | Result |
| 120 | 5 |
| 24 | 4 |
| 362880 | 9 |

# PROGRAM:

# n=int(input())

# a=1

# for i in range(1,n+1):

# a \*= i

# print(a)

# 

# Ex. No. : 4.2 Date:

# Register No.: Name:

# [Non](https://www.rajalakshmicolleges.net/moodle/mod/quiz/view.php?id=5717) Repeated Digit Count

# Write a program to find the count of non-repeated digits in a given number N. The number will be passed to the program as an input of type int.

# Assumption: The input number will be a positive integer number >= 1 and <= 25000.

# Some examples are as below.

# If the given number is 292, the program should return 1 because there is only 1 non-­repeated digit '9' in this number

# If the given number is 1015, the program should return 2 because there are 2 non-repeated digits in this number, '0', and '5'.

# If the given number is 108, the program should return 3 because there are 3 non-­repeated digits in this number, '1', '0', and '8'.

# If the given number is 22, the function should return 0 because there are NO non-­repeated digits in this number.

# 

# For Example:

| Input | Result |
| --- | --- |
| 292 | 1 |
| 1015 | 2 |
| 108 | 3 |
| 22 | 0 |

# PROGRAM:

# num=input()

# C=0

# for digit in set(num):

# if num.count(digit) == 1:

# c+=1

# print(c)

# 

# Ex. No. : 4.3 Date:

# Register No.: Name:

# Prime Checking

# Write a program that finds whether the given number N is Prime or not. If the number is prime, the program should return 2 else it must return 1.

# Assumption: 2 <= N <=5000, where N is the given number.

# Example1: if the given number N is 7, the method must return 2

# Example2: if the given number N is 10, the method must return 1

# For example:

| Input | Result |
| --- | --- |
| 7 | 2 |
| 10 | 1 |

# PROGRAM:

# N=int(input())

# if N<2:

# result =1

# else:

# for i in range(2,int(N \*\* 0.5)+1):

# if N%i == 0:

# result=1

# break

# else:

# result=2

# print(result)

# 

# Ex. No. : 4.4 Date:

# Register No.: Name:

# Next Perfect Square

# Given a number N, find the next perfect square greater than N.

# Input Format:

# Integer input from stdin.

# Output Format:

# Perfect square greater than N.

# Example Input:

# 10

# Output:

# 16

# PROGRAM:

# import math

# N=int(input())

# sqrt\_N=math. sqrt(N)

# next\_int=math.ceil(sqrt\_N)

# next\_perfect\_square=next\_int \*\* 2

# print(next\_perfect\_square)

# 

# Ex. No. : 4.5 Date:

# Register No.: Name:

# Nth Fibonacci

# Write a [program](https://www.rajalakshmicolleges.net/moodle/mod/quiz/view.php?id=3478) to return the nth number in the fibonacci series. The value of N will be passed to the [program](https://www.rajalakshmicolleges.net/moodle/mod/quiz/view.php?id=3478) as input.

# NOTE: Fibonacci series looks like –

# 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, . . . and so on.

# i.e. Fibonacci series starts with 0 and 1, and continues generating the next number as the sum of the previous two numbers.

# • first Fibonacci number is 0,

# • second Fibonacci number is 1,

# • third Fibonacci number is 1,

# • fourth Fibonacci number is 2,

# • fifth Fibonacci number is 3,

# • sixth Fibonacci number is 5,

# • seventh Fibonacci number is 8, and so on.

# For example:

# Input:

# 7

# Output

# 8

# PROGRAM:

# n=int(input())

# a,b=0,1

# for x in range(n-1):

# a,b=b,a+b

# print(a)

# 

# Ex. No. : 4.6 Date:

# Register No.: Name:

# Write a program to find the count of the number of prime numbers in a specified range.

# The starting and ending number of the range will be provided as input to the program.

# Assumption: 2 <= starting number of the range <= ending number of the range <= 7919

# Example1: If the starting and ending number or the range is given as 2 and 20, the program

# must return 8, because there are 8 prime numbers in the specified range from 2 to 20.

# namely (2. 3. 5, 7, 11, 13, 17, 19)

# Example2: If the starting and ending number of the range is given as 700 and 725, the

# program must return 3, because there are 3 prime numbers in the specified range from 700

# to 725, namely (701, 709, 719)

# For example:

|  |  |
| --- | --- |
| Input | Result |
| 220 | 8 |
| 700725 | 3 |

# PROGRAM:

# a=int(input())

# b=int(input())

# C=0

# for num in range(a,b+1):

# if num>1:

# for i in range(2,int(num \*\* 0.5)+1):

# if (num%i) == 0:

# break

# else:

# C+=1

# print(c)

# 

# Ex. No. : 4.7 Date:

# Register No.: Name:

# An abundant number is a number for which the sum of its proper divisors is greater than the

# number itself.

# Proper divisors of the number are those that are strictly lesser than the number.

# Input Format:

# Take input an integer from stdin

# Output Format:

# Print Yes if given number is Abundant. Otherwise, print No

# Example input:

# 12

# Output:

# Yes

# Explanation

# The proper divisors of 12 are: 1, 2, 3, 4, 6, whose sum is 1 + 2 + 3 + 4 + 6 = 16. Since sum of

# proper divisors is greater than the given number, 12 is an abundant number.

# Example input:

# 13

# Output:

# No

# Explanation

# The proper divisors of 13 is: 1, whose sum is 1. Since sum of proper divisors is not greater

# than the given number, 13 is not an abundant number.

# PROGRAM:

# N=int(input())

# sum\_divisors=0

# for i in range(1,N):

# if N%i == 0:

# sum\_divisors+=i

# if sum\_divisors>N:

# print("Yes")

# else:

# print("No")

# 

# Ex. No. : 4.8 Date:

# Register No.: Name:

# Given a positive integer N, check whether it can be represented as a product of single digit

# numbers.

# Input Format:

# Single Integer input.

# Output Format:

# Output displays Yes if condition satisfies else prints No.

# Example Input:

# 14

# Output:

# Yes

# Example Input:

# 13

# Output:

# No

# n=int(input())

# can\_be\_product=False

# for i in range(2,10):

# if n%i == 0 and n//i<10:

# can\_be\_product=True

# break

# if can\_be\_product:

# print("Yes")

# else:

# print("No")

# 

# Ex. No. : 4.9 Date:

# Register No.: Name:

# Product of single digit

# Given a positive integer N, check whether it can be represented as a product of single digit numbers.

# Input Format:

# Single Integer input.

# Output Format:

# Output displays Yes if condition satisfies else prints No.

# Example Input:

# 14

# Output:

# Yes

# Example Input:

# 13

# Output:

# No

# PROGRAM:

# n=int(input())

# can\_be\_product=False

# for i in range(2,10):

# if n%i == 0 and n//i<10:

# can\_be\_product=True

# break

# if can\_be\_product:

# print("Yes")

# else:

# print("No")

# 

# Ex. No. : 4.10 Date:

# Register No.: Name:

# Perfect Square After adding One

# Given an integer N, check whether N the given number can be made a perfect square after adding 1 to it.

# Input Format:

# Single integer input.

# Output Format:

# Yes or No.

# Example Input:

# 24

# Output:

# Yes

# Example Input:

# 26

# Output:

# No

# For example:

| Input | Result |
| --- | --- |
| 24 | Yes |

# PROGRAM:

# import math

# num=int(input())

# sqrt\_num=math.isqrt(num+1)

# if num+1 == sqrt\_num\*sqrt\_num:

# print("Yes")

# else:

# print("No")

