Given a non-negative integer x, return the square root of x rounded down to the nearest integer. The returned integer should be non-negative as well. You must not use any built-in exponent function or operator.

Example 1:

Input: x = 4 Output: 2 Explanation: The square root of 4 is 2, so we return 2.

Example 2:

Input: x = 8 Output: 2 Explanation: The square root of 8 is 2.82842..., and since we round it down to the nearest integer, 2 is returned.

Constraints:

0 <= x <= 2^31 – 1

ANS –

To solve this problem, we can use the binary search approach to find the square root of a non-negative integer. Here is the step-by-step process to solve the problem:

Handle base cases:

If x is 0 or 1, return x as the square root.

Set the left and right boundaries for the binary search:

Set left = 0 and right = x.

Perform binary search:

While left <= right, do the following steps:

Calculate the middle value as (left + right) / 2.

Calculate the square of the middle value as mid\_squared = middle \* middle.

If mid\_squared is equal to x, return middle as the square root.

If mid\_squared is less than x, update the left boundary to middle + 1.

If mid\_squared is greater than x, update the right boundary to middle - 1.

Return the result:

If no perfect square root is found, return right (the floor value of the square root).

Here is the Python implementation of the above steps:

def mySqrt(x):

# Handle base cases

if x == 0 or x == 1:

return x

# Binary search

left = 0

right = x

while left <= right:

middle = (left + right) // 2

mid\_squared = middle \* middle

if mid\_squared == x:

return middle

elif mid\_squared < x:

left = middle + 1

else:

right = middle - 1

# Return the floor value of the square root

return right

# Test cases

print(mySqrt(4)) # Output: 2

print(mySqrt(8)) # Output: 2

This implementation will find the square root of the given non-negative integer x and return the result rounded down to the nearest integer.