Debugging problems day 1

1. 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.

   For example, do not use pow(x, 0.5) in c++ or x ** 0.5 in python.
   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.

```
class Solution {
      int mySqrt(int x) {
      }
}
```

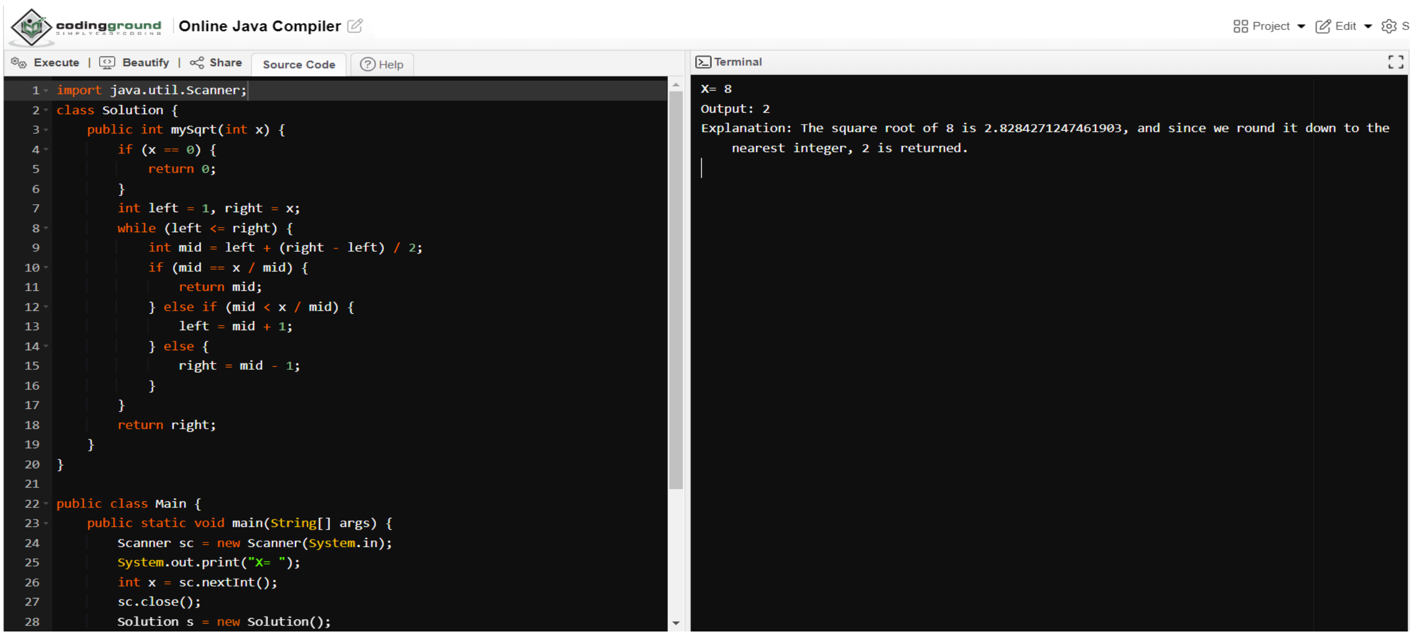
DEBUGGING CODE:

```
import java.util.Scanner;
class Solution {
   public int mySqrt(int x) {
      if (x == 0) {
         return 0;
      }
      int left = 1, right = x;
      while (left <= right) {
         int mid = left + (right - left) / 2;
         if (mid == x / mid) {
            return mid;
         } else if (mid < x / mid) {
            left = mid + 1;
         } else {
            right = mid - 1;
         }
      }
```

```
        return right;
    }
}

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("X= ");
        int x = sc.nextInt();
        sc.close();
        Solution s = new Solution();
        int result = s.mySqrt(x);
        System.out.println("Output: " + result);
        System.out.println("Explanation: The square root of " + x + " is " +
Math.sqrt(x) +
                ", and since we round it down to the nearest integer, " + result
+ " is returned.");
    }
}
```

Output:



```
import java.util.Scanner;
class Solution {
    public int mySqrt(int x) {
        if (x == 0) {
            return 0;
        }
        int left = 1, right = x;
        while (left <= right) {
            int mid = left + (right - left) / 2;
            if (mid == x / mid) {
                return mid;
            } else if (mid < x / mid) {
                left = mid + 1;
            } else {
                right = mid - 1;
            }
        }
        return right;
    }
}

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("X= ");
        int x = sc.nextInt();
        sc.close();
        Solution s = new Solution();
```

```
X= 8
Output: 2
Explanation: The square root of 8 is 2.8284271247461903, and since we round it down to the
    nearest integer, 2 is returned.
```

2. **Given an integer x, return true if x is a palindrome , and false otherwise.**

Example 1:
Input: x = 121
Output: true
Explanation: 121 reads as 121 from left to right and from right to left.
Example 2:
Input: x = -121
Output: false
Explanation: From left to right, it reads -121. From right to left, it becomes 121-. Therefore it is not a palindrome.
Example 3:
Input: x = 10
Output: false
Explanation: Reads 01 from right to left. Therefore it is not a palindrome.

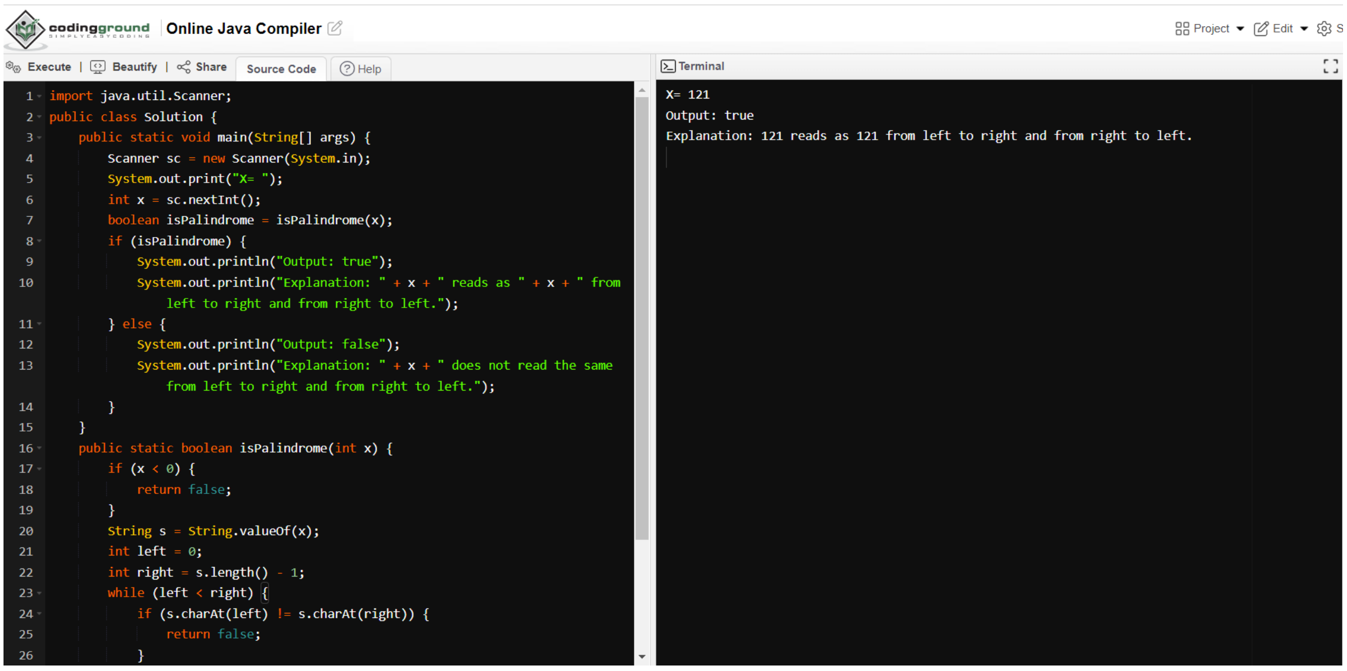
```
class Solution {
    bool isPalindrome(int x) {
    }
 }
```

DEBUGGING CODE:

```
import java.util.Scanner;
public class Solution {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    System.out.print("X= ");
    int x = sc.nextInt();
    boolean isPalindrome = isPalindrome(x);
    if (isPalindrome) {
      System.out.println("Output: true");
      System.out.println("Explanation: " + x + " reads as " + x + " from left to right and from
right to left.");
    } else {
      System.out.println("Output: false");
      System.out.println("Explanation: " + x + " does not read the same from left to right
and from right to left.");
    }
  }
  public static boolean isPalindrome(int x) {
    if (x < 0) {
      return false;
    }
    String s = String.valueOf(x);
    int left = 0;
```

```
        int right = s.length() - 1;
        while (left < right) {
            if (s.charAt(left) != s.charAt(right)) {
                return false;
            }
            left++;
            right--;
        }
        return true;
    }
}
```

Output:



```
import java.util.Scanner;
public class Solution {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("X= ");
        int x = sc.nextInt();
        boolean isPalindrome = isPalindrome(x);
        if (isPalindrome) {
            System.out.println("Output: true");
            System.out.println("Explanation: " + x + " reads as " + x + " from
                left to right and from right to left.");
        } else {
            System.out.println("Output: false");
            System.out.println("Explanation: " + x + " does not read the same
                from left to right and from right to left.");
        }
    }
    public static boolean isPalindrome(int x) {
        if (x < 0) {
            return false;
        }
        String s = String.valueOf(x);
        int left = 0;
        int right = s.length() - 1;
        while (left < right) {
            if (s.charAt(left) != s.charAt(right)) {
                return false;
            }
```

```
X= 121
Output: true
Explanation: 121 reads as 121 from left to right and from right to left.
```