**Assignment 3**

**Reverse first half of a number:**

import java.util.Scanner;

public class ReverseFirstHalf {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter a number: ");

int num = scanner.nextInt();

int digits = 0;

int temp = num;

while (temp > 0) {

digits++;

temp /= 10;

}

int divisor = 1;

for (int i = 1; i <= digits / 2; i++) {

divisor \*= 10;

}

int firstHalf = num / divisor;

int secondHalf = num % divisor;

int reversedFirstHalf = 0;

while (firstHalf > 0) {

int digit = firstHalf % 10;

reversedFirstHalf = reversedFirstHalf \* 10 + digit;

firstHalf /= 10;

}

int reversedNum = reversedFirstHalf \* divisor + secondHalf;

System.out.println("Reversed number: " + reversedNum);

}

}

**2. Check if reverse of a number is a perfect square:**

import java.util.Scanner;

public class ReverseAndPerfectSquare {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter a number: ");

int num = scanner.nextInt();

int reversedNum = 0;

while (num > 0) {

int digit = num % 10;

reversedNum = reversedNum \* 10 + digit;

num /= 10;

}

double sqrt = Math.sqrt(reversedNum);

if (sqrt - (int)sqrt == 0) {

System.out.println(reversedNum + " is a perfect square.");

} else {

System.out.println(reversedNum + " is not a perfect square.");

}

}

}

**3. Print Armstrong numbers between a range:**

public class ArmstrongNumbers {

public static void main(String[] args) {

int start = 1, end = 50;

for (int num = start; num <= end; num++) {

int originalNum = num;

int sum = 0, digits = 0;

while (originalNum > 0) {

digits++;

originalNum /= 10;

}

originalNum = num;

while (originalNum > 0) {

int digit = originalNum % 10;

sum += Math.pow(digit, digits);

originalNum /= 10;

}

if (sum == num) {

System.out.print(num + " ");

}

}

}

}

**4. Print prime numbers between a range:**

public class PrimeNumbers {

public static void main(String[] args) {

int start = 1, end = 50;

for (int num = start; num <= end; num++) {

if (isPrime(num)) {

System.out.print(num + " ");

}

}

}

public static boolean isPrime(int num) {

if (num <= 1) {

return false;

}

for (int i = 2; i \* i <= num; i++) {

if (num % i == 0) {

return false;

}

}

return true;

}

}

**5. Print nth prime number:**

public class NthPrimeNumber {

public static void main(String[] args) {

int n = 3; // Change this to the desired n

int count = 0, num = 2;

while (count < n) {

if (isPrime(num)) {

count++;

}

num++;

}

System.out.println(nth + " prime number is: " + (num - 1));

}

// Same isPrime method as above

}

**6. Check if a number is prime without using a count:**

public class PrimeNumberCheck {

public static void main(String[] args) {

int num = 11;

if (isPrime(num)) {

System.out.println(num + " is a prime number.");

} else {

System.out.println(num + " is not a prime number.");

}

}

// Same isPrime method as above

}

**7. Print perfect numbers between a range:**

public class PerfectNumbers {

public static void main(String[] args) {

int start = 1, end = 50;

for (int num = start; num <= end; num++) {

int sum = 0;

for (int i = 1; i < num; i++) {

if (num % i == 0) {

sum += i;

}

}

if (sum == num) {

System.out.print(num + " ");

}

}

}

}

**8. Print reverse Fibonacci series:**

public class ReverseFibonacci {

public static void main(String[] args) {

int n = 10;

int a = 0, b = 1, c = 0;

while (c <= n) {

c = a + b;

a = b;

b = c;

}

while (c > 0) {

c = a - b;

a = b;

b = c;

if (c >= 0) {

System.out.print(c + " ");

}

}

}

}

**9. Check for co-prime numbers:**

public class CoPrimeNumbers {

public static void main(String[] args) {

int num1 = 4, num2 = 5;

if (gcd(num1, num2) == 1) {

System.out.println(num1 + " and " + num2 + " are co-prime.");

} else {

System.out.println(num1 + " and " + num2 + " are not co-prime.");

}

}

public static int gcd(int a, int b) {

if (b == 0) {

return a;

}

return gcd(b, a % b);

}

}