## LAB_3\generators_present_in_Zn.py

```python

# 21CSE1003
# Ashish Singh

# Q1. Write a program to find the list of generators present in Zn* where n is a large integer.

from math import gcd

def find_generators(n):
  generators = []

  zn_star = [x for x in range(1, n) if gcd(x, n) == 1]

  for g in zn_star:
    powers = [g]
    while powers[-1] != 1:
      powers.append((powers[-1] * g) % n)
    if len(powers) == len(zn_star):
      generators.append(g)

  return generators

n = int(input("Enter n: "))
generators = find_generators(n)
print("Generators of Zn* for n =", n, ":", generators)
```

**LAB_3\cyclic_group_present_in_range.py**

```python

# Q2. Write a program to find the list of cyclic group present within a range (Example: 2000 to 3000).

from math import gcd

def is_primitive_root(g, n):
    required_set = set(num for num in range(1, n) if gcd(num, n) == 1)
    actual_set = set(pow(g, powers, n) for powers in range(1, n))
    return required_set == actual_set

def find_cyclic_groups(start, end):
    cyclic_groups = []

    for n in range(start, end + 1):
        if n == 1:  # The group Z1 is trivial and not considered cyclic
            continue
        if any(is_primitive_root(g, n) for g in range(1, n)):
            cyclic_groups.append(n)

    return cyclic_groups

start = 200
end = 300
cyclic_groups = find_cyclic_groups(start, end)
print(cyclic_groups)

```

## LAB_3\order_of_element_in_ZnStar.py

```python

# Q3. Write a program to find the order of an element in Zn where n is a large integer.

from math import gcd

def order_of_element(g, n):
    if gcd(g, n) != 1:
        return None  # g must be coprime with n to belong to Zn*

    k = 1
    power = g % n
    while power != 1:
        power = (power * g) % n
        k += 1

    return k

n = int(input("Enter n: "))
g = int(input("Enter g: "))
order = order_of_element(g, n)
if order:
    print(f"The order of element {g} in Z_{n}* is {order}.")
else:
    print(f"Element {g} is not in Z_{n}* (not coprime with {n}).")

```

**LAB_3\quadratic_residue_nonresidue.py**

```python

# Q4. Write a program to find the quadratic residue and quadratic nonresidue mod n where n is a large integer.

from math import gcd

def Zn_star(n):
    Zn = [i for i in range(n)]

    Zn_ = [] # this is Zn*
    for i in Zn:
        if gcd(i, n) == 1:
            Zn_.append(i)
    print(f"Zn* = {Zn_}")

    return Zn_


def Qn_Qn_bar(n):
    Zn_ = Zn_star(n)
    Qn = []
    for i in Zn_:
        Qn.append(i**2 % n)
    Qn = set(Qn)
    Zn_ = set(Zn_)
    Qn_bar = Zn_ - Qn
    print(f"Qn = {Qn}")
    print(f"Qn_bar = {Qn_bar}")

n = int(input("\nEnter value of n: "))

Qn_Qn_bar(n)
```

**LAB_3\square_root_modulo_n.py**

```python

# Q5. Write a program to find the square root of a modulo n where n is a large integer.

from math import gcd

def Zn_star(n):
    Zn = [i for i in range(n)]

    Zn_ = [] # this is Zn*
    for i in Zn:
        if gcd(i, n) == 1:
            Zn_.append(i)
    return Zn_


def root_modulo_n(n, a):
    Zn_ = Zn_star(n)
    s = []
    for i in Zn_:
        if i**2 % n == a:
            s.append(i)
    print(s)


n = int(input("\nEnter value of n: "))
a = int(input("Enter value of a: "))

root_modulo_n(n, a)

```