7.1

```
import math

def shanks_algorithm(alpha, beta, p):
# Step 1: Initialize
m = math.ceil(math.sqrt(p))
# Step 2: Baby-step
baby_steps = {}
for i in range(m):
value = pow(alpha, i, p)
baby_steps[value] = i
# Step 3: Giant-step
inv_alpha_m = pow(alpha, -m, p)
giant_step = beta
for j in range(m):
if giant_step in baby_steps:
i = baby_steps[giant_step]
return j * m + i
giant_step = (giant_step * inv_alpha_m) % p
return None

# Test cases from the problem
alpha1, beta1, p1 = 106, 12375, 24691
alpha2, beta2, p2 = 6, 248388, 458009

log1 = shanks_algorithm(alpha1, beta1, p1)
log2 = shanks_algorithm(alpha2, beta2, p2)

print(log1, log2)
```

Result:(解释部分在文字）

```
(base) wangyidan@wangyidandeMacBook-Pro 密码学 % python -u "/Users/wangyidan/Deskt
op/密码学/HW(7.1).py"
22392 232836
```

7.9

```
import sympy

# 定义参数
p = 31847
alpha = 5
a = 7899

# 定义一个函数进行解密
def elgamal_decrypt(c1, c2, p, a):
s = pow(c1, a, p)
```

```
s_inv = sympy.mod_inverse(s, p)
m = (c2 * s_inv) % p
return m

# 示例密文对
ciphertext_pairs = [(3781, 14409)]

# 解密示例
for c1, c2 in ciphertext_pairs:
m = elgamal_decrypt(c1, c2, p, a)
print(f"Decrypted number: {m}")

# 解码为字母
first_letter = chr((m // (26**2)) + ord('A'))
second_letter = chr(((m // 26) % 26) + ord('A'))
third_letter = chr((m % 26) + ord('A'))
print(f"Decrypted text: {first_letter}{second_letter}{third_letter}")
```

Result:（解释部分在文字）

```
(base) wangyidan@wangyidandeMacBook-Pro 密码学 % python -u "/Users/wangyidan/Deskt
op/密码学/HW(7.9).py"
Decrypted number: 12354
Decrypted text: SHE
```

7.23

```
import random

class FiniteGroup:
def __init__(self, order):
self._order = order
self._generator = random.randint(2, order - 1)
def order(self):
return self._order
def generator(self):
return self._generator
def random_element(self):
return random.randint(2, self._order - 1)

class OracleDDH:
def __init__(self, G):
self.G = G
def is_DDH_tuple(self, g, g_a, g_b, g_ab):
# Simulated DDH oracle logic for this example
a = pow(g_a, 1, self.G.order())
b = pow(g_b, 1, self.G.order())
return pow(g, a * b, self.G.order()) == g_ab
```

```
class ElGamal:
def __init__(self, G, alpha, beta):
self.G = G
self.alpha = alpha
self.beta = beta
def encrypt(self, x):
k = random.randint(1, self.G.order() - 1)
y1 = pow(self.alpha, k, self.G.order())
y2 = (x * pow(self.beta, k, self.G.order())) % self.G.order()
return (y1, y2)

class OracleDISTINGUISH:
def __init__(self, G):
self.G = G
def distinguish(self, x1, x2, y1, y2):
# Simulated distinguishing logic for this example
return y2 in [(x1 * y1) % self.G.order(), (x2 * y1) % self.G.order()]

def distinguish_ElGamal_encryptions(x1, x2, y1, y2, oracle_ddh):
g = oracle_ddh.G.generator()
beta = oracle_ddh.G.random_element()
is_DDH_1 = oracle_ddh.is_DDH_tuple(g, y1, beta, y2 * pow(x1, -1, oracle_ddh.G.order()))
is_DDH_2 = oracle_ddh.is_DDH_tuple(g, y1, beta, y2 * pow(x2, -1, oracle_ddh.G.order()))
if is_DDH_1:
return True
elif is_DDH_2:
return True
else:
return False

def solve_DDH(g, g_a, g_b, g_ab, oracle_distinguish):
x1 = 1
x2 = 2
y1 = g_a
y2_1 = g_ab
y2_2 = (g_ab * pow(g_a, -1, oracle_distinguish.G.order())) % oracle_distinguish.G.order()
if oracle_distinguish.distinguish(x1, x2, y1, y2_1):
return True
elif oracle_distinguish.distinguish(x1, x2, y1, y2_2):
return False
else:
return False

# Example usage
```

```
order = 1019 # A prime number for the order of the group
G = FiniteGroup(order)
alpha = G.generator()
beta = G.random_element()
oracle_ddh = OracleDDH(G)
elgamal = ElGamal(G, alpha, beta)

x1 = random.randint(1, order - 1)
x2 = random.randint(1, order - 1)
y1, y2 = elgamal.encrypt(x1)

result_distinguish = distinguish_ElGamal_encryptions(x1, x2, y1, y2, oracle_ddh)
print(f"ElGamal encryption distinguishing result: {result_distinguish}")

oracle_distinguish = OracleDISTINGUISH(G)

g = G.generator()
a = random.randint(1, order - 1)
b = random.randint(1, order - 1)
g_a = pow(g, a, G.order())
g_b = pow(g, b, G.order())
g_ab = pow(g, a * b, G.order())

result_solve_ddh = solve_DDH(g, g_a, g_b, g_ab, oracle_distinguish)
print(f"Decision Diffie-Hellman result: {result_solve_ddh}")
```

Result:(解释部分在文字)

```
(base) wangyidan@wangyidandeMacBook-Pro 密码学 % python -u "/Users/wangyidan/Deskt
op/密码学/HW(7.23)_1.py"
ElGamal encryption distinguishing result: False
Decision Diffie-Hellman result: False
```