Q1 Team Name
0 Points

Goldfish

Q2 Commands
10 Points

List the commands used in the game to reach the ciphertext

go,enter,pick,c,back,give,back,back,thrnxxtzy,read

Q3 Analysis
50 Points

Give a detailed analysis of how you figured out the password? (Explain in less than 500 words)

After giving above commands , we got hints in form of group theory equations, which we solved in following ways to get the password.

prime modulus p = 455470209427676832372575348833

we had three equation in form of q*g^a , where q is password, g is a element of the group and a is an integer

Given

$q * g^{429} = 431955503618234519808008749742 mod(p)$ -----eqn(1)

$q * g^{1973} = 176325509039323911968355873643 mod(p)$ -----eqn(2)

$q * g^{7596} = 98486971404861992487294722613 mod(p)$ ------eqn(3)

Assignment 3

GRADED

GROUP

Alok Kumar Trivedi

Akshay Kumar Chittora

Jeet Sarangi

View or edit group

TOTAL POINTS

70 / 70 pts

QUESTION 1

Team Name **0** / 0 pts

QUESTION 2

Commands **10** / 10 pts

QUESTION 3

Analysis **50** / 50 pts

QUESTION 4

Password **10** / 10 pts

QUESTION 5

we will refer $q * g^a$ term in each equation as x_1, x_2, x_3 respectively

Dividing eqn(2) by eqn(1):

$g^{1544} = x_2 * x_1^{-1} (\mathrm{mod} p)$ -----eqn(4)

Dividing eqn(3) by eqn(2):

$g^{5623} = x_3 * x_2^{-1} (\mathrm{mod} p)$ -----eqn(5)

Dividing eqn(3) by eqn(1):

$g^{7167} = x_3 * x_1^{-1} (\mathrm{mod} p)$ -----eqn(6)

To perform above operation we need to get the multiplicative inverses of x_1, x_2 . as g is member of the multiplicative group , so x_1, x_2 must also be members of the group and since p is prime x_1, x_2 are coprime to p. Hence their multiplicative inverse also exist.

let y_1 be the inverse of x_1

Then , $(x_1 * y_1) \bmod p = 1$

since p is a prime number , we can use Fermat little theorem to get the inverse

$x_1^{p-1} \equiv 1 (\mathrm{mod} p)$

multiplying both side by x_1^{-1}

$x_1^{p-2} \equiv x_1^{-1} (\mathrm{mod} p) = 707499967902234717329046816 40 (\mathrm{mod} p)$

from eqn(4)

$g^{1544} = 70749996790223471732904681640 (\mathrm{mod} p)$ -----eqn(7)

similarly by solving eqn(5) and eqn(6) we get

$g^{5623} = 42041307425102202802727078 5553 (\mathrm{mod} p)$ -----eqn(8)

$g^{7167} = 11041137667091891262690752 6185 (\mathrm{mod} p)$ -----eqn(9)

Multiplying both side of eqn(9) by inverse of $(g^{1544})^3$

$g^{7167} * ((g^{1544})^4)^{-1} (\mathrm{mod} p) \equiv g^{991} =$
$16179855827055696173242482 2635 \bmod p$

Similarly

$(g^{991})^2 * ((g^{1544})^4)^{-1} (\mathrm{mod} p) \equiv g^{438} =$
$32759748229808211969556819 2760 (\mathrm{mod} p)$

$(g^{991}) * ((g^{438})^2)^{-1} (\mathrm{mod} p) \equiv g^{115} =$
$21242776032541733631689363 8262 (\mathrm{mod} p)$

$(g^{115})^4 * ((g^{438})^1)^{-1}(\text{mod}p) \equiv g^{22} =$
$628758645601568765677831278 11(\text{mod}p)$

$(g^{22})^2 1 * ((g^{115})^4)^{-1}(\text{mod}p) \equiv g^2 =$
$1080449076654660139356277860 69(\text{mod}p)$

$(g^{115})^1 * ((g^{22})^5)^{-1}(\text{mod}p) \equiv g^5 =$
$2546621559808707232733340225 69(\text{mod}p)$

$(g^5)^1 * ((g^2)^2)^{-1}(\text{mod}p) \equiv g = 525650854179633110276943 39(\text{mod}p)$

It is mentioned in the hints that g is 5____50__4______31______94__9 , which actually matches with our answer.

Now we determine q (password) by putting value of g in eqn(1)

which is $q * g^{429} = 431955503618234519808008749742 mod(p)$

We multiply both side of eqn(1) by $(g^{429})^{-1}$,

$q = ((g^{429})^{-1} * 431955503618234519808008749742) mod(p)$

The multiplicative inverse of (g^{429}) is 442956820316148690889301696615

That gives us value of q = password as 134721542097659029845273957

Hence the password is 134721542097659029845273957

Q4 Password

10 Points

What was the final command used to clear this level?

134721542097659029845273957

Q5 Codes

0 Points

Upload any code that you have used to solve this level

Assignment 3.ipynb Download

In [25]:

```
def gcd(x,y):
    if(x==0):
        return y
    return gcd(y%x,x)
```

In [26]:

```
def po(a,b,m):
    if(b==0):
        return 1

    p = po(a,b // 2,m)%m
    p = (p*p)%m

    if(b % 2 == 0):
        return p
    else:
        return ((a*p) % m)
```

In [32]:

```
def mI(a , m):
    g = gcd(a, m)
    if(g == 1):
        y = po(a, m-2 ,m)
        print("The modular inverse is ",y)
        return y
    else:
        print("Does not exist")
```

In [33]:

```
y1 = 4319555036182345198080087497‌42
y2 = 1763255090393239119683558736‌43
y3 = 984869714048619924872947226‌13
p = 455470209427676832372575348‌833
```

In [34]:

```
y1_inverse = mI(y1,p)
y2_inverse = mI(y2,p)
```

```
The modular inverse is  707499967902234717329046816‌40
The modular inverse is  228947149478752602606353685‌125
```

In [35]:

```
g_1544 = (y2 * y1_inverse)%p
g_1544
```

Out [35]:

```
111590994894663139264552154672
```

In [36]:

```
g_7167 = (y3 * y1_inverse)%p
g_7167
```

Out [36]:

```
1104113766709189126269075261 85
```

In [37]:

```
g_5623 = (y3 * y2_inverse)%p
g_5623
```

Out [37]:

```
420413074251022028027270785553
```

In [38]:

```
g_991 = (g_7167 * mI(po(g_1544,4,p),p))%p
g_991
```

```
The modular inverse is  304296090672599420401986286302
```

Out [38]:

```
161798558270556961732424822635
```

In [39]:

```
g_438 = (po(g_991,2,p) * mI(g_1544,p))%p
g_438
```

```
The modular inverse is  218173384175465437436454958180
```

Out [39]:

```
327597482298082119695568192760
```

In [40]:

```
g_115 = (mI(po(g_438,2,p),p) * g_991)%p
g_115
```

```
The modular inverse is  297374246948059676278983181681
```

Out [40]:

```
212427760325417336316893638262
```

In [41]:

```
g_22 = (mI(g_438,p) * po(g_115,4,p))%p
g_22
```

```
The modular inverse is  453530656410176241507046342872
```

Out [41]:

```
62875864560156876567783127811
```

In [42]:

```
g_2 = (mI(po(g_115,4,p),p) * po(g_22,21,p))%p
g_2
```

The modular inverse is 105600931401330644523752113862

Out [42]: 108044907665466013935627786069

In [43]:
```
g_5 = (mI(po(g_22,5,p),p)*g_115)%p
g_5
```

The modular inverse is 100551735247242729663164535176

Out [43]: 254662155980870723273334022569

In [44]:
```
g = (mI(po(g_2,2,p),p) * g_5)%p
g
```

The modular inverse is 254950689434017345885415339945

Out [44]: 525650854179633110276943339

In [45]:
```
password = (y1*mI(po(g,429,p),p))%p
password
```

The modular inverse is 442956820316148690889301696615

Out [45]: 1347215420976590298452739557