ASSIGNMENT 1
CS674: POST QUANTUM SECURITY

Name:- VIKRANT CHAUHAN
Roll No. :- 231110407
Batch:- Y23
Programe:- MS(R) CYBERSECURITY

1.Question

Write a software code for performing negative wrapped convolution using Number Theoretic Transformation (NTT) for the parameters $n = 512$, $q = 12289$, $g = 10968$ for $Rq = Zq[X]/Xn + 1$. Your code must have three distinct functions:

- A function that will take the input in Rq and will convert the input into its corresponding NTT transformation using radix-2 Cooley-Tukey method.
- A function that will take the two inputs which are already transformed by NTT and will produce their point wise multiplication
- A function that will take the inputs in NTT transformed domain and will perform inverse NTT using the radix-2 Gentleman-Sande method.

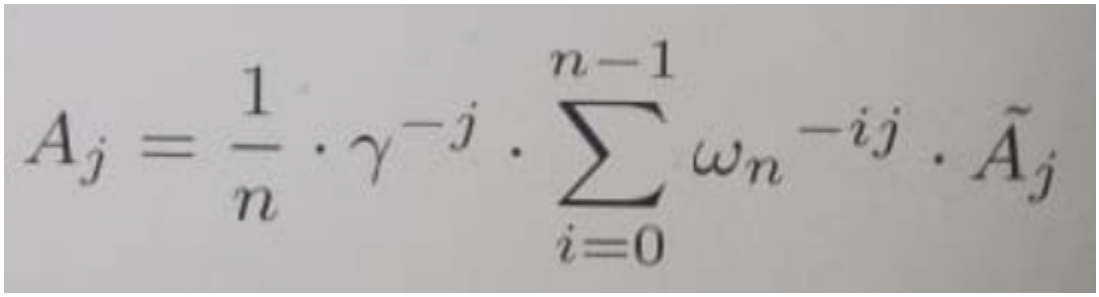
2.Important Formulas

->We have to write code for negative wrapped convolution using Number Theoretic Transformation (NTT).The important formulas are:-

$$\tilde{A}_j = \sum_{i=0}^{n-1} \gamma^i \cdot \omega_n{}^{ij} \cdot A_i$$

->The above formula is used to calculate Negative Wrapped NTT.

->For PointWise Multiplication we just have to do component-wise multiplication this means we Suppose have a=[a0,a1,a2…….] and b=[b0,b1,b2…….].Then the pointwise multiplication of a and b is c=[a0*b0,a1*b1,a2*b2,.....]

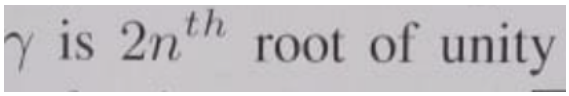


$$A_j = \frac{1}{n} \cdot \gamma^{-j} \cdot \sum_{i=0}^{n-1} \omega_n{}^{-ij} \cdot \tilde{A}_j$$

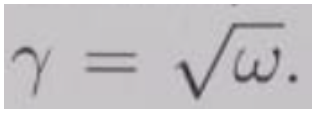
->The above formula is used to calculate Negative Wrapped Inverse NTT.

->Actually if we directly calculate the NTT and Inverse NTT with the help of the above formula then the Time Complexity would beO(n^2).But we have better algorithms using Cooley-Tukey method and Gentleman-Sande Method with the help of which we can calculate the results in O(nLog(n)) Time.

->Some Important Variables and Formulas used:-

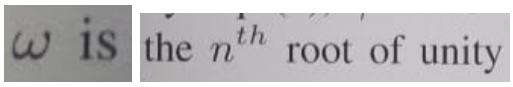


γ is $2n^{th}$ root of unity



$$\gamma = \sqrt{\omega}.$$

$$\gamma^{2n} \equiv 1 \mod q$$



ω is the n^{th} root of unity

3.Algorithm Of My Code

->We firstly take 2 inputs of size 512 namely P_N1 and P_N2.I have made them randomly generated inputs of size 512.These inputs cannot have a value greater than q.

```
P_N1=np.random.randint(0, q, n)
```

```
P_N2=np.random.randint(0, q, n)
```

->We firstly need parameters for my calculation.The parameters are:-

```
n=512

q=12289

gamma=10968

omega_HH=(gamma**2)%q
```

->Now before doing NTT operation I have created a gamma array which will be required during the NTT calculation

```
make_gamma_array()

def make_gamma_array():

  for i in range(0,n):

     temp=pow(gamma,i,q)

     gamma_array[i]=temp
```

->Now actually we find out the NTT of PN_1 and P_N2

```
P_N1_Modified1=NTT(t1)

P_N2_Modified1=NTT(t2)
```

->Now I do negative wrapped NTT calculation using the below code.The Below is NTT transformation using radix-2 Cooley-Tukey method The Time complexity of the below code is O(Nlogn) :-

```
def NTT(P_NN):
  P_temp_Modified=P_NN
  t = n
  m=1
  gamma_original_ntt=Bitreverse(gamma_array,math.log2(n))
  while(m<n):
     t = int(t/2)
     for i in range(0,m):
        j1 =2*i*t
        j2 = j1 + t - 1
        S = gamma_original_ntt[m + i]
        for j in range(j1,j2+1):
           U = P_temp_Modified[j]
           V =P_temp_Modified[j+t]*S
           P_temp_Modified[j]=(U+V)%q
           P_temp_Modified[j+t]=(U-V)%q
     print(P_temp_Modified)
```

```
    m=m*2

  return P_temp_Modified
```

->Now after finding the NTT we actually have to pass the two obtained NTT outputs to PointWise Multiplication Function

```
Product_Result=PointWise_Multiplication(P_N1_Modified1,P_N2_Modified1)
```

->Now I calculate the PointWise Multiplication with the help of the below code

```
def PointWise_Multiplication(NTT_Poly1,NTT_Poly2):
  result=[]
  for i in range(0,len(NTT_Poly1)):
    temp=(NTT_Poly1[i]*NTT_Poly2[i])%q
    result.append(temp)
  return result
```

->Now we have to calculate the inverse NTT but before doing that we form the gamma inverse array

```
make_gamma_inverse_array()
def make_gamma_inverse_array():
  gamma_inv=gamma_inverse_calculate(gamma)
  for i in range(0,n):
    temp=pow(gamma_inv,i,q)
    t=reverse_num_binary(i)
    gamma_inverse_reverse[t]=temp
```

->Now actually we find out the negative wrapped INTT

```
P_Final=Inverse_NTT(Product_Result)
```

->Now I do negative wrapped Inverse NTT calculation using the below code.The Below is NTT transformation using radix-2 Gentleman-Sande method The Time complexity of the below code is O(Nlogn) :-

```
def Inverse_NTT(P_Modified):

  P_temp_Modified2=P_Modified

  t = 1

  m=n

  while(m>1):

     j1 = 0

     h = int(m/2)

     for i in range(0,h):

        j2 = j1 + t- 1

        G = gamma_inverse_reverse[h + i]

        j=j1



        while(j<=j2):

           u1 = P_temp_Modified2[j]

           t1 =P_temp_Modified2[j+t]

           P_temp_Modified2[j]=(u1+t1)%q
```

```
            P_temp_Modified2[j + t] = ((u1-t1)*G)%q

            j+=1

        j1 = j1 + 2*t

    t = 2*t

    m=int(m/2)

for i in range(0,n):

    P_temp_Modified2[i]=(P_temp_Modified2[i]*n_inverse(n))%q

return P_temp_Modified2
```

->The Result Of the negative wrapped Inverse NTT is our final answer.

```
P_Final=Inverse_NTT(Product_Result)

print("After INTT ",P_Final)
```

4.Verification

->To check whether our output is coming correct or not we know that the result which we obtained after negative wrapped Inverse NTT should be equal to multiplication of polynomials P_N1 and P_N2 and then take mod x^N+1.If these two results are equal then our answer is correct.

->For verification I have written the Below code which is used to find the multiplication of PN1 &PN_2 and then take mod x^N+1.

```
print("Verification :)")

f = np.zeros(n + 1)
```

```
f[0] = 1

f[n] = 1

ans=np.remainder(np.polydiv(np.polymul(P_N1[::-1],
P_N2[::-1]),f)[1],q).astype(int)[::-1]

print(ans)
```

->Now we check equate ans and P_Final.If the result comes out as True then our result obtained is correct.In my code i am getting TRUE for all the random values of P_N1 and P_N2.

5.Output ScreenShots

```
(vikrantchauhan@Vikrants-MacBook-Air Assignment % python3 Vikrant_Neg_Wrap.py
PN1 is  [ 3815  3133  9194  2774  4112  2277  9465 11354  5130  5607 10764  5908
   352 11134  5576  8973  1913 11462  9623  9348 11770  3679  6207  2478
   630  8255   302  4825  3917  4495 11998  9296 10661  2214  4506  9050
  5891 12248  1515  3376  9561 11590  4595 12058 10264  9901  1036  2186
  4517  5570  1084  6989  4166  3487  2822  1245  1738 11170  9329  5661
  1082  3243  2183  5463  5874 10902  6871  1299  6345  2728 10053  1350
  1792  9758  1499 12276  8396  5442 11097  5096  3534   128 10117  1719
   180 10845  7038  9371  6856  8175  2666   866  7757  1618 12136  6995
  6539  7486  6779   615   522  5215  4199  5806  9129  6114  8494  8431
  3844  5689  6034 12041  4433  2613  9251  2387  6940  2201  9917   673
  6063  3170  4857 10972 11438  5167  5051   668  5449   628  7992  1591
  6800  5196  2589  3277 10831   397  5373  3747 11965 10786  1443  7752
  9560  6588  8023  6308  9344  7668 11268  5877  7973   413 11381  5635
  3740  4201  1243   809  1754  5807  6750 11358 11478  7281 10394 10216
  5032  7975  8832  8050  6624   314  8931 11008  7778  4428  9292   914
 11443  3777  7853  8101 11900  1395 12244 11785  1624   293 10041   194
  3964  2490  7595  3133  2908  6503  9106  5702  2675  9599  2078  9494
  2523  7736  8176 12098 11431 11297  7517  5380 11369  1758 11464  1531
  6305  9341  5571  3117  7863   416  3278  9507  1729  9471  6914  3306
  2310 12148  8652  5227  1998  1276   799 10548 11964   733  2183  4005
 12107  9971 11290  7266  4822  5771 10782  8938   744  4042  8617 12171
  8730  1339  6353   343  4217  4560  3940  9900  3829  9241  6313  3564
  6033  1387  2986 12282 10796  6341  9585  7546  6936  5341  3148 11766
 11905 12098  2105  4946  1101  2340  9143  1793 11286  5583  6591  7636
  6532  5990 10934  7535 10205  4224  8010  8328  2953  8435  7236  1777
   511  3937  5544  8600  5732 11702   567 11363  8161  8270  7017 10673
  2876   683 10313   332  6098  7898  7248  5064   966 12198  9283  7641
  9479  4955  6681  5601  4929   422 11960 12261  7526  3827  3946  7905
  5713 10528  5706  7022  3798  9287  9625  2184   798  1611 11895 11319
 11383  1322 10582  2296  4539  4185  2193 10672  3677  3588  9393  7632
  2346   398  2181  7187  6837   789  4320  1682 11503  5473  5266  7669
  6634  9876  6924  4368  9812  2959  8837  3983 10064 11719 12009  4917
  8589  1336  6905  2746 10832  9847 10316  6300 10572  5353  7501  4986
  5004 10776  3057  1994 10896  7182   552  4685  8670 11099  5247 11468
  9039  6644  8669  7963  3539  8359 11456  1051  2685 11827  5830  1490
 11304  7865  2349   266 12130  3528  8434  4777  1343  6998 10137 10294
 12089  8875   732  3989  1651  2563  9193  4037 10764  8518  9536 10212
  2056  2402  7226 10144   558  5944  8616  2437  9476  6865  7130  8426
  8704  1979  6520  4226 10430  5120 12242  2192   266  9964  6021   105
  2124  5619   172   771 12062    93   177  2731   309   770   783  5027
  2523 11341  8016  4186 11275  2596 11477  8611  2481  3398  4126  2241
  2807  1311  9750  2718  9499  7354  3699 10622 10603  1318   432  4848
  3782  7238  1593 10982  4634  7744  4536  4236]
PN2 is  [ 1705  1009  2375  3749 11453  4744  5948   189 12287   386 10104 10078
  5642  8814  4441  2999 11311  3069  5642  2567  2342 11852  2578  6837
  3804  2075  5311   397  8587  4935  1378  3733  9180  8439  9058  7165
  8812 10650 11472  6251  4064  4479  3308  3454  5234  1949  6451 10031
  1338   819  1974  2071  7999  9308  2219  2322  8948  1996  4202  6236
  8441  5233  5782  3975  3528 11913  9382  2842  1253   480 10681  1025
  5950  9517 10910  7651  5384  7504 11882 10038  1893   776  1411  6004
  6354  5110  6122  7501  7954  6311  3988  1533  4108  7831  3862  1034
  5804   578  6085  6409  8106  2087  3026 10624   141  4340  2460  7784
 11423  6239  3220 12136  8033  4432 11190  1994  2093   414   368  3612
 11116  1159  6133  2272  9712   533 10830  6638  8269  9636  6046  8355
```

Figure : Inputs PN1 And PN2

```
Gamma array  [1, 10968, 3, 8326, 9, 400, 27, 1200, 81, 3600, 243, 10800, 729, 7822, 2187, 11177, 6561, 8953,
1, 9928, 9764, 5206, 4714, 3329, 1853, 9987, 5559, 5383, 4388, 3860, 875, 11580, 2625, 10162, 7875, 5908, 1
468, 8646, 7404, 1360, 9923, 4080, 5191, 12240, 3284, 12142, 9852, 11848, 4978, 10966, 2645, 8320, 7935, 38
5, 3704, 10327, 11112, 6403, 8758, 6920, 1696, 8471, 5088, 835, 2975, 2505, 8925, 7515, 2197, 10256, 6591,
734, 7969, 4624, 11618, 1583, 10276, 4749, 6250, 1958, 6461, 5874, 7094, 5333, 8993, 3710, 2401, 11130, 720
8076, 10745, 11939, 7657, 11239, 10682, 9139, 7468, 2839, 10115, 8517, 5767, 973, 5012, 2919, 2747, 8757, 8
10806, 5092, 7840, 2987, 11231, 8961, 9115, 2305, 2767, 6915, 8301, 8456, 325, 790, 975, 2370, 2925, 7110,
099, 11287, 8719, 9283, 1579, 3271, 4737, 9813, 1922, 4861, 5766, 2294, 5009, 6882, 2738, 8357, 8214, 493,
8, 6234, 10805, 6413, 7837, 6950, 11222, 8561, 9088, 1105, 2686, 3315, 8058, 9945, 11885, 5257, 11077, 3482
42, 453, 3748, 1359, 11244, 4077, 9154, 12231, 2884, 12115, 8652, 11767, 1378, 10723, 4134, 7591, 113, 1048
59, 11973, 11899, 11341, 11119, 9445, 8779, 3757, 1759, 11271, 5277, 9235, 3542, 3127, 10626, 9381, 7300, 3
2912, 11994, 8736, 11404, 1630, 9634, 4890, 4324, 2381, 683, 7143, 2049, 9140, 6147, 2842, 6152, 8526, 6167
10963, 6608, 8311, 7535, 355, 10316, 1065, 6370, 3195, 6821, 9585, 8174, 4177, 12233, 242, 12121, 726, 1178
0040, 9280, 5542, 3262, 4337, 9786, 722, 4780, 2166, 2051, 6498, 6153, 7205, 6170, 9326, 6221, 3400, 6374,
, 1207, 3123, 3621, 9369, 10863, 3529, 8011, 10587, 11744, 7183, 10654, 9260, 7384, 3202, 9863, 9606, 5011,
```

Figure : Gamma Array

Figure : Result Of NTT of PN1 and PN2

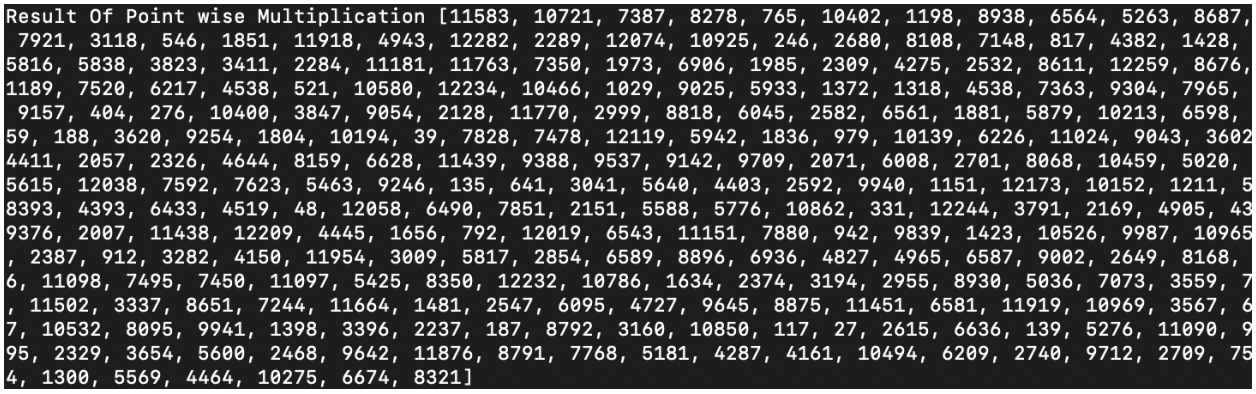


Figure : Result Of PointWise Multiplication

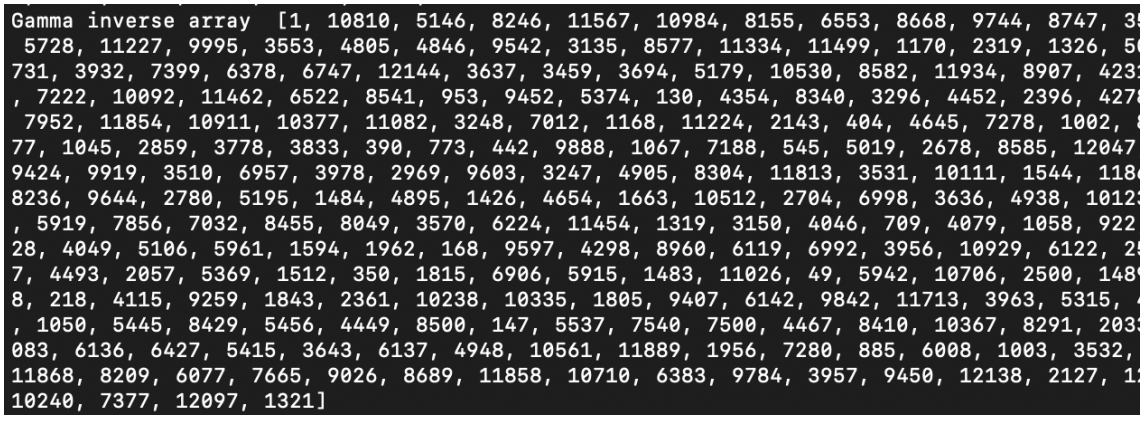


Figure : Gamma Inverse Array

```
Result Of INTT Final Answer  [11875, 11205, 8771, 10171, 5756, 9266, 519, 6984, 12081, 7732, 9
7, 6055, 2005, 11218, 10781, 2031, 4712, 7814, 7474, 5601, 1610, 1851, 4373, 9016, 7720, 5770,
10823, 3715, 1465, 5782, 9008, 93, 3955, 5679, 9474, 7879, 6423, 6468, 818, 1835, 10974, 3889,
668, 4824, 7633, 2458, 9700, 385, 9824, 2841, 9164, 10684, 9573, 6927, 8597, 1761, 8015, 12259
 600, 6368, 5863, 2750, 2219, 6590, 7494, 865, 618, 8244, 4749, 12250, 2470, 8618, 9708, 3537,
4, 9852, 3709, 2472, 8259, 11874, 4909, 4747, 7804, 7752, 8398, 2288, 3392, 1614, 651, 4525, 5
99, 5957, 5423, 9699, 5902, 6455, 4780, 3193, 439, 9717, 4046, 10765, 1585, 15, 11116, 587, 60
7, 5331, 10169, 9317, 9697, 2343, 7285, 11221, 12231, 1821, 480, 8024, 11733, 4862, 8174, 5723
49, 6828, 4572, 5432, 8931, 9583, 2848, 2065, 1191, 2949, 11938, 3391, 4245, 7503, 5845, 4579,
, 8866, 7713, 5461, 9596, 2420, 6121, 6422, 11003, 2381, 10219, 2338, 11754, 11973, 5469, 3270
768, 6861, 5126, 5613, 9615, 4428, 10996, 3114, 1494, 10893, 7535, 1955, 8408, 10471, 2444, 23
5, 1629, 4350, 3983, 12011, 10712, 268, 8418, 7662, 6073, 2742, 4681, 637, 10624, 924, 4165, 1
17, 2332, 11819, 1638, 10682, 3050, 5430, 4228, 1566, 9658, 670, 6349, 9580, 6204, 4766, 10374
, 7465, 4212, 10222, 1151, 2412, 1050, 6694, 8920, 9623, 7151, 329, 6118, 4789, 11375, 1932, 5
 2055, 1989, 8648, 3958, 1782, 1523, 9032, 1183, 11524, 5773, 8079, 4821, 5284, 2124, 49, 1068
4, 8889, 5358]
Verification :)
```

Figure : Result Of INTT

VERIFICATION STARTS:-

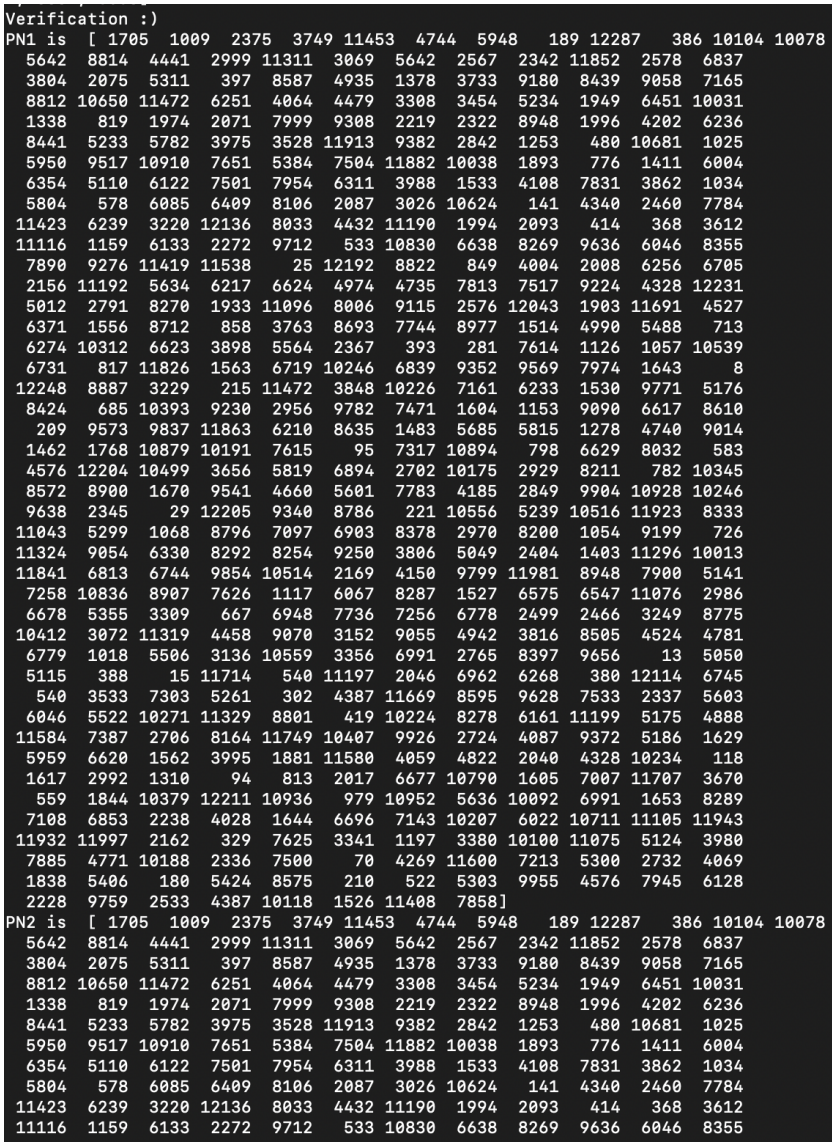


Figure : Inputs PN1 And PN2

```
Polynomial Multiplication and modulo  [11875 11205  8771 10171  5756  9266
 12173  1429  3898 12143 11492  5640  5191  2058  4566  2105  2213  7033
  1833  6520 11943 10730  6457  6055  2005 11218 10781  2031  4712  7814
  7474  5601  1610  1851  4373  9016  7720  5770 12019  5374  8134  4992
  1034  1011  2597  9164   119 11911  1231  1345  1004  1808  5617 11003
     2  7072  2034 10823  3715  1465  5782  9008    93  3955  5679  9474
  7879  6423  6468   818  1835 10974  3889  9379   654  6755  5382  6608
  8031  4540 11085  7924  7460  5768   643   334  8576  7026  3921  6410
 11323   408  3668  4824  7633  2458  9700   385  9824  2841  9164 10684
  9573  6927  8597  1761  8015 12259  4144  7213  6402  4348  4912  7460
  5979  7701  8256   937  1011  5055 10626  4941  3682  5033  7641  6977
   603   600  6368  5863  2750  2219  6590  7494   865   618  8244  4749
 12250  2470  8618  9708  3537  1897  6442 10737 11909  3891  8739  6398
 12029  9802  6040  9800  2344   591  6572  3102 10608   863  2800  9794
  9852  3709  2472  8259 11874  4909  4747  7804  7752  8398  2288  3392
  1614   651  4525  5488 10729  4805  1374  7036  4534 11293  9411  1296
  4829 11883  8448 11602  5945  4910  4522  4751 11405  4799  5957  5423
  9699  5902  6455  4780  3193   439  9717  4046 10765  1585    15 11116
   587  6060   207  9249  1654  8014  8206  3555  2591  3811  9333  1326
 10781  9578   156  8635  1904  7466  8955  8417  1697  5331 10169  9317
  9697  2343  7285 11221 12231  1821   480  8024 11733  4862  8174  5723
   915 11491  4026 11979  5644  2783  9559  3927  2643   728  5325  1659
  2486  2966 10388  6884 10059  8895  1549  6828  4572  5432  8931  9583
  2848  2065  1191  2949 11938  3391  4245  7503  5845  4579  2146 12203
  4804  5548  5823  9957  2438   516 11063  2533  9455  9860  7169  8214
  9632  7132  9816  1593  7608  8866  7713  5461  9596  2420  6121  6422
 11003  2381 10219  2338 11754 11973  5469  3270  8855  8063  8334  2192
  4633  9057  9334  8345  6996  7852  4322 10849  2806  7197 12137 10205
  5822  1275  6768  6861  5126  5613  9615  4428 10996  3114  1494 10893
  7535  1955  8408 10471  2444  2382 10610  2079  1542  8668  1676  7835
  4318  9119  1640  5235 11747 11242  6219 10734  6574  7317  5908 11985
  1629  4350  3983 12011 10712   268  8418  7662  6073  2742  4681   637
 10624   924  4165 11208  9300  6492  7887  1469  3906 11608    33  9436
  4196  7103  8707 10747  3699   792  5613  3206  1479   221  7417  2332
 11819  1638 10682  3050  5430  4228  1566  9658   670  6349  9580  6204
  4766 10374 10450  6081  4884  8492 10463   786 11226 11247  1341 12133
  9607 12118  7559  7101  7997 10530  9177  2564  7465  4212 10222  1151
  2412  1050  6694  8920  9623  7151   329  6118  4789 11375  1932  5689
   462  5383 10044 10208  6114  8292  2167  5830    51  5928  6423  6095
    66  8073   563  9147  1519  9195  4934  2055  1989  8648  3958  1782
  1523  9032  1183 11524  5773  8079  4821  5284  2124    49 10681   105
  2839 10693  8659   767  7738 11774  5730  6203  8514  2633  6348  5368
  6415  1575 10967   412  1992  2774  8889  5358]
```

Figure : Output Of Polynomial Multiplication And Modulo X^n+1

```
   462  5383 10044 10208  6114  8292  2167  5830    51  5928  6423  6095
    66  8073   563  9147  1519  9195  4934  2055  1989  8648  3958  1782
  1523  9032  1183 11524  5773  8079  4821  5284  2124    49 10681   105
  2839 10693  8659   767  7738 11774  5730  6203  8514  2633  6348  5368
  6415  1575 10967   412  1992  2774  8889  5358]
True
Your Answer Is Correct
vikrantchauhan@Vikrants-MacBook-Air Assignment %
```

Figure : Final Verification Output

->We can see that the final output is coming as TRUE and Your Answer is Correct is Printed Therefore my answer is correct.

->Hence Our Result is successfully Verified.Hence Proved

6.References

1. CS674 Class Notes
2. Algorithm Given In Assignment PDF
3. Speeding up the Number Theoretic Transform for Faster Ideal Lattice-Based CryptographyPatrick Longa and Michael Naehrig,Microsoft Research, USA
4. Compact Ring-LWE Cryptoprocessor Sujoy Sinha Roy1, Frederik Vercauteren1, Nele Mentens1, Donald Donglong Chen2 and Ingrid Verbauwhede1
5. High-Performance Ideal Lattice-Based Cryptography on 8-bit ATxmega Microcontrollers Extended Version Thomas P¨oppelmann, Tobias Oder, and Tim Gu¨neysu

Name:- VIKRANT CHAUHAN
Roll No. :- 231110407
Batch:- Y23
Programe:- MS(R) CYBERSECURITY