PROGRAMMING ASSIGNMENT 1 DOCUMENTATION

Submitted By:- Amal Majeed Mucheth Abdulmajeed
200415928

This program was written in **Python** programming language to be executed with python interpreter version **2.7**. The aim of the program is to implement two functions PRGA and inverse PRGA(IPRGA) of RC4 stream cipher encryption to move from one 256 bit RC4 state (S_n,i_n,j_n) to the next 256 bit state (S_{n+1},i_{n+1},j_{n+1}) with PRGA algorithm and vice versa for IPRGA algorithm.

Implementation

The function **PRGA** takes in 4 arguments :

1) **S** : - the 256 bit random stream array used for encryption, here we implement it as an array of size 256 with values from 0 - 255 populated randomly to emulate the Key Scheduling Algorithm result.
2) i - 8 bit index pointer for swapping procedure to further randomise the array S
3) j - 8 bit index pointer for swapping procedure to further randomise the array S
4) n - number of times the PRGA algorithm has to be applied on initial state (S,i,j)

The function I**PRGA** takes in 4 arguments :

5) **S** : - the 256 bit random resultant array from PRGA implementation on the initial S after n steps.
6) i - 8 bit index pointer value at the end of n steps of PRGA on S
7) j - 8 bit index pointer value at the end of n steps of PRGA on S
8) n - number of times the IPRGA algorithm has to be applied on Final state (S_n,i_n,j_n) to obtain the initial state (S,i,j)

Working

```
Initial Random state 'S' after KSA


[191, 149, 23, 222, 146, 235, 148, 106, 31, 187, 86, 236, 27, 239, 220, 98, 104,
 211, 112, 226, 93, 13, 215, 214, 234, 81, 118, 224, 35, 116, 199, 4, 190, 142,
249, 22, 150, 168, 71, 8, 253, 247, 141, 225, 217, 84, 78, 195, 18, 228, 108, 62
, 103, 15, 102, 28, 20, 147, 44, 114, 242, 174, 11, 26, 90, 1, 77, 46, 36, 49, 1
34, 136, 231, 68, 194, 16, 218, 244, 152, 241, 138, 109, 210, 227, 192, 162, 121
, 67, 196, 47, 193, 51, 245, 167, 113, 59, 53, 182, 89, 233, 55, 216, 74, 200, 7
0, 212, 173, 170, 30, 185, 61, 65, 133, 7, 105, 10, 169, 88, 25, 145, 206, 111,
223, 2, 73, 255, 229, 159, 171, 180, 125, 76, 54, 14, 219, 75, 129, 237, 95, 64,
 39, 131, 92, 172, 99, 42, 139, 41, 57, 72, 183, 132, 100, 203, 189, 60, 157, 52
, 43, 140, 45, 107, 178, 164, 97, 66, 117, 209, 85, 143, 82, 230, 248, 135, 205,
 3, 254, 6, 176, 128, 58, 181, 207, 201, 179, 119, 96, 250, 80, 197, 5, 144, 155
, 123, 37, 166, 175, 17, 213, 48, 110, 87, 21, 202, 126, 160, 154, 63, 29, 208,
127, 115, 251, 163, 156, 50, 9, 204, 12, 33, 151, 221, 122, 252, 24, 56, 79, 38,
 161, 243, 19, 158, 232, 0, 91, 69, 186, 177, 32, 240, 40, 188, 124, 130, 83, 16
5, 120, 153, 198, 101, 246, 34, 137, 184, 238, 94]
```

Initially an array S will be initialized using python's **random library** to emulate implementation of KSA algorithm before PRGA. Next the 'i' and 'j' pointer values to be swapped will be entered by the programmer along with 'n' the number of steps :

```
Enter the number of steps to proceed along PRGA : 4
Enter the current state of pointer 'i' : 0
Enter the current state of pointer 'j'  : 0
```

Next , the swapping procedure begins 'n' number of times and if the programmer wants to see the intermediate states after swapping , keep pressing 'y' at the prompt when asked , if pressed 'n' then the algorithm directly takes to the end state (S_n,i_n,j_n) without further prompting :

```
Do you want to see next state ?(PRESS y - Yes , n - No)  y
Indices swapped :  1  &  149

[191, 72, 23, 222, 146, 235, 148, 106, 31, 187, 86, 236, 27, 239, 220, 98, 104, 211, 112, 226, 93, 13, 215, 2
14, 234, 81, 118, 224, 35, 116, 199, 4, 190, 142, 249, 22, 150, 168, 71, 8, 253, 247, 141, 225, 217, 84, 78,
195, 18, 228, 108, 62, 103, 15, 102, 28, 20, 147, 44, 114, 242, 174, 11, 26, 90, 1, 77, 46, 36, 49, 134, 136,
 231, 68, 194, 16, 218, 244, 152, 241, 138, 109, 210, 227, 192, 162, 121, 67, 196, 47, 193, 51, 245, 167, 113
, 59, 53, 182, 89, 233, 55, 216, 74, 200, 70, 212, 173, 170, 30, 185, 61, 65, 133, 7, 105, 10, 169, 88, 25, 1
45, 206, 111, 223, 2, 73, 255, 229, 159, 171, 180, 125, 76, 54, 14, 219, 75, 129, 237, 95, 64, 39, 131, 92, 1
72, 99, 42, 139, 41, 57, 149, 183, 132, 100, 203, 189, 60, 157, 52, 43, 140, 45, 107, 178, 164, 97, 66, 117,
209, 85, 143, 82, 230, 248, 135, 205, 3, 254, 6, 176, 128, 58, 181, 207, 201, 179, 119, 96, 250, 80, 197, 5,
144, 155, 123, 37, 166, 175, 17, 213, 48, 110, 87, 21, 202, 126, 160, 154, 63, 29, 208, 127, 115, 251, 163, 1
56, 50, 9, 204, 12, 33, 151, 221, 122, 252, 24, 56, 79, 38, 161, 243, 19, 158, 232, 0, 91, 69, 186, 177, 32,
240, 40, 188, 124, 130, 83, 165, 120, 153, 198, 101, 246, 34, 137, 184, 238, 94]



Do you want to see next state ?(PRESS y - Yes , n - No)  y
Indices swapped :  2  &  172

[191, 72, 248, 222, 146, 235, 148, 106, 31, 187, 86, 236, 27, 239, 220, 98, 104, 211, 112, 226, 93, 13, 215,
214, 234, 81, 118, 224, 35, 116, 199, 4, 190, 142, 249, 22, 150, 168, 71, 8, 253, 247, 141, 225, 217, 84, 78,
 195, 18, 228, 108, 62, 103, 15, 102, 28, 20, 147, 44, 114, 242, 174, 11, 26, 90, 1, 77, 46, 36, 49, 134, 136
, 231, 68, 194, 16, 218, 244, 152, 241, 138, 109, 210, 227, 192, 162, 121, 67, 196, 47, 193, 51, 245, 167, 11
3, 59, 53, 182, 89, 233, 55, 216, 74, 200, 70, 212, 173, 170, 30, 185, 61, 65, 133, 7, 105, 10, 169, 88, 25,
145, 206, 111, 223, 2, 73, 255, 229, 159, 171, 180, 125, 76, 54, 14, 219, 75, 129, 237, 95, 64, 39, 131, 92,
172, 99, 42, 139, 41, 57, 149, 183, 132, 100, 203, 189, 60, 157, 52, 43, 140, 45, 107, 178, 164, 97, 66, 117,
 209, 85, 143, 82, 230, 23, 135, 205, 3, 254, 6, 176, 128, 58, 181, 207, 201, 179, 119, 96, 250, 80, 197, 5,
144, 155, 123, 37, 166, 175, 17, 213, 48, 110, 87, 21, 202, 126, 160, 154, 63, 29, 208, 127, 115, 251, 163, 1
56, 50, 9, 204, 12, 33, 151, 221, 122, 252, 24, 56, 79, 38, 161, 243, 19, 158, 232, 0, 91, 69, 186, 177, 32,
240, 40, 188, 124, 130, 83, 165, 120, 153, 198, 101, 246, 34, 137, 184, 238, 94]
```

Towards the end of the algorithm , the final state (S_n,i_n,j_n) will be printed out to the programmer and the program enters IPRGA logic to reverse the effects of PRGA on the array S.

```
Do you want to see next state ?(PRESS y - Yes , n - No)  n
Indices swapped :  3  &  138

Indices swapped :  4  &  28


Random state 'S' after PRGA


[191, 72, 248, 95, 35, 235, 148, 106, 31, 187, 86, 236, 27, 239, 220, 98, 104, 211, 112, 226, 93, 13, 215, 21
4, 234, 81, 118, 224, 146, 116, 199, 4, 190, 142, 249, 22, 150, 168, 71, 8, 253, 247, 141, 225, 217, 84, 78,
195, 18, 228, 108, 62, 103, 15, 102, 28, 20, 147, 44, 114, 242, 174, 11, 26, 90, 1, 77, 46, 36, 49, 134, 136,
 231, 68, 194, 16, 218, 244, 152, 241, 138, 109, 210, 227, 192, 162, 121, 67, 196, 47, 193, 51, 245, 167, 113
, 59, 53, 182, 89, 233, 55, 216, 74, 200, 70, 212, 173, 170, 30, 185, 61, 65, 133, 7, 105, 10, 169, 88, 25, 1
45, 206, 111, 223, 2, 73, 255, 229, 159, 171, 180, 125, 76, 54, 14, 219, 75, 129, 237, 222, 64, 39, 131, 92,
172, 99, 42, 139, 41, 57, 149, 183, 132, 100, 203, 189, 60, 157, 52, 43, 140, 45, 107, 178, 164, 97, 66, 117,
 209, 85, 143, 82, 230, 23, 135, 205, 3, 254, 6, 176, 128, 58, 181, 207, 201, 179, 119, 96, 250, 80, 197, 5,
144, 155, 123, 37, 166, 175, 17, 213, 48, 110, 87, 21, 202, 126, 160, 154, 63, 29, 208, 127, 115, 251, 163, 1
56, 50, 9, 204, 12, 33, 151, 221, 122, 252, 24, 56, 79, 38, 161, 243, 19, 158, 232, 0, 91, 69, 186, 177, 32,
240, 40, 188, 124, 130, 83, 165, 120, 153, 198, 101, 246, 34, 137, 184, 238, 94]
```

Inside the IPRGA logic , again the programmer is prompted whether he/she wants to see the intermediate array while swapping and works exactly like PRGA but all the operations are opposite in nature are order in order to reverse or inverse the effects of PRGA on S :-

```
                          IPRGA Logic


Do you want to see next state ?(PRESS y - Yes , n - No)  y
Indices swapped :  4  &  28

[191, 72, 248, 95, 146, 235, 148, 106, 31, 187, 86, 236, 27, 239, 220, 98, 104, 211, 112, 226, 93, 13, 215, 2
14, 234, 81, 118, 224, 35, 116, 199, 4, 190, 142, 249, 22, 150, 168, 71, 8, 253, 247, 141, 225, 217, 84, 78,
195, 18, 228, 108, 62, 103, 15, 102, 28, 20, 147, 44, 114, 242, 174, 11, 26, 90, 1, 77, 46, 36, 49, 134, 136,
 231, 68, 194, 16, 218, 244, 152, 241, 138, 109, 210, 227, 192, 162, 121, 67, 196, 47, 193, 51, 245, 167, 113
, 59, 53, 182, 89, 233, 55, 216, 74, 200, 70, 212, 173, 170, 30, 185, 61, 65, 133, 7, 105, 10, 169, 88, 25, 1
45, 206, 111, 223, 2, 73, 255, 229, 159, 171, 180, 125, 76, 54, 14, 219, 75, 129, 237, 222, 64, 39, 131, 92,
172, 99, 42, 139, 41, 57, 149, 183, 132, 100, 203, 189, 60, 157, 52, 43, 140, 45, 107, 178, 164, 97, 66, 117,
 209, 85, 143, 82, 230, 23, 135, 205, 3, 254, 6, 176, 128, 58, 181, 207, 201, 179, 119, 96, 250, 80, 197, 5,
144, 155, 123, 37, 166, 175, 17, 213, 48, 110, 87, 21, 202, 126, 160, 154, 63, 29, 208, 127, 115, 251, 163, 1
56, 50, 9, 204, 12, 33, 151, 221, 122, 252, 24, 56, 79, 38, 161, 243, 19, 158, 232, 0, 91, 69, 186, 177, 32,
240, 40, 188, 124, 130, 83, 165, 120, 153, 198, 101, 246, 34, 137, 184, 238, 94]


Do you want to see next state ?(PRESS y - Yes , n - No)  y
Indices swapped :  3  &  138

[191, 72, 248, 222, 146, 235, 148, 106, 31, 187, 86, 236, 27, 239, 220, 98, 104, 211, 112, 226, 93, 13, 215,
214, 234, 81, 118, 224, 35, 116, 199, 4, 190, 142, 249, 22, 150, 168, 71, 8, 253, 247, 141, 225, 217, 84, 78,
 195, 18, 228, 108, 62, 103, 15, 102, 28, 20, 147, 44, 114, 242, 174, 11, 26, 90, 1, 77, 46, 36, 49, 134, 136
, 231, 68, 194, 16, 218, 244, 152, 241, 138, 109, 210, 227, 192, 162, 121, 67, 196, 47, 193, 51, 245, 167, 11
3, 59, 53, 182, 89, 233, 55, 216, 74, 200, 70, 212, 173, 170, 30, 185, 61, 65, 133, 7, 105, 10, 169, 88, 25,
145, 206, 111, 223, 2, 73, 255, 229, 159, 171, 180, 125, 76, 54, 14, 219, 75, 129, 237, 95, 64, 39, 131, 92,
172, 99, 42, 139, 41, 57, 149, 183, 132, 100, 203, 189, 60, 157, 52, 43, 140, 45, 107, 178, 164, 97, 66, 117,
 209, 85, 143, 82, 230, 23, 135, 205, 3, 254, 6, 176, 128, 58, 181, 207, 201, 179, 119, 96, 250, 80, 197, 5,
144, 155, 123, 37, 166, 175, 17, 213, 48, 110, 87, 21, 202, 126, 160, 154, 63, 29, 208, 127, 115, 251, 163, 1
56, 50, 9, 204, 12, 33, 151, 221, 122, 252, 24, 56, 79, 38, 161, 243, 19, 158, 232, 0, 91, 69, 186, 177, 32,
```

And as long as the programmer keeps pressing ‘y’ at the prompt , he/she will be able to see the intermediate snapshots of the array S while the IPRGA algorithm works for ‘n’ steps.

```
Do you want to see next state ?(PRESS y - Yes , n - No)  n
Indices swapped :  2  &  172

Indices swapped :  1  &  149


Random state 'S' after IPRGA


[191, 149, 23, 222, 146, 235, 148, 106, 31, 187, 86, 236, 27, 239, 220, 98, 104, 211, 112, 226, 93, 13, 215,
214, 234, 81, 118, 224, 35, 116, 199, 4, 190, 142, 249, 22, 150, 168, 71, 8, 253, 247, 141, 225, 217, 84, 78,
 195, 18, 228, 108, 62, 103, 15, 102, 28, 20, 147, 44, 114, 242, 174, 11, 26, 90, 1, 77, 46, 36, 49, 134, 136
, 231, 68, 194, 16, 218, 244, 152, 241, 138, 109, 210, 227, 192, 162, 121, 67, 196, 47, 193, 51, 245, 167, 11
3, 59, 53, 182, 89, 233, 55, 216, 74, 200, 70, 212, 173, 170, 30, 185, 61, 65, 133, 7, 105, 10, 169, 88, 25,
145, 206, 111, 223, 2, 73, 255, 229, 159, 171, 180, 125, 76, 54, 14, 219, 75, 129, 237, 95, 64, 39, 131, 92,
172, 99, 42, 139, 41, 57, 72, 183, 132, 100, 203, 189, 60, 157, 52, 43, 140, 45, 107, 178, 164, 97, 66, 117,
209, 85, 143, 82, 230, 248, 135, 205, 3, 254, 6, 176, 128, 58, 181, 207, 201, 179, 119, 96, 250, 80, 197, 5,
144, 155, 123, 37, 166, 175, 17, 213, 48, 110, 87, 21, 202, 126, 160, 154, 63, 29, 208, 127, 115, 251, 163, 1
56, 50, 9, 204, 12, 33, 151, 221, 122, 252, 24, 56, 79, 38, 161, 243, 19, 158, 232, 0, 91, 69, 186, 177, 32,
240, 40, 188, 124, 130, 83, 165, 120, 153, 198, 101, 246, 34, 137, 184, 238, 94]
```

By looking at the intermediate results or the indices swapped, the programmer can easily **verify that the states can move forward by the PRGA algorithm and those states can be retraced or reversed back to a previous state by the IPRGA algorithm** in the same manner.

Dependencies / Software Requirements

- **Python 2.7** interpreter (On how to install if not already installed , visit : https://www.python.org/downloads/)

Execution

The command environment varies from system to system , the below specifications are for linux based operating systems (Ubuntu , MacOS etc) and might be different for windows systems.

1. In the command prompt type in the path to the interpreter followed by the name of the program file as follows : -

```
Amals-MacBook-Air:prog_assgmnt amalmajeed$ /usr/bin/python PA1.py
```

Where **PA1.py** is the name of the program and **/usr/bin/python** is the full system path for the python2.7 interpreter (this varies from system to system).