QUESTION 5(a) AND 5(b) REPORT

NAME:PRANJALI BISHNOI
ROLL NUMBER:2021101038
SECTION: B

PART 5(a):

CASE 1:
We iterate from i=1 to i=127 and generate 0 or 1 randomly using the random function, the probability of getting a 0 or 1 at index i is obviously 1/2 as both are equally likely.

CASE 2:
We iterate from i=128 to i=1000000, where ($x_i = x(i-1) \oplus x(i-127)$ for $i \geq 128$)

Probability of getting 0 at index i= [(probability of 0 at index (i-1)) * (probability of 0 at index (i-127))] + [(probability of 1 at index (i-1)) * (probability of 1 at index (i-127))]

$$= [(1/2) * (1/2)] + [(1/2) * (1/2)]$$
$$= (1/4) + (1/4)$$
$$= (1/2)$$

Probability of getting 1 at index i= [(probability of 0 at index (i-1)) * (probability of 0 at index (i-127))] + [(probability of 1 at index (i-1)) * (probability of 1 at index (i-127))]

$$= [(1/2) * (1/2)] + [(1/2) * (1/2)]$$
$$= (1/4) + (1/4)$$
$$= (1/2)$$

Next, we generate 0 or 1 randomly using the random function (rand()%2 approach), the probability of getting a 0 or 1 at index i is obviously 1/2 as both are equally likely.

CODE:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main()
{
    int count0=0, count1=0, rand0=0, rand1=0, array[1000000];
    srand(time(NULL));

    for (int x = 1; x <= 127; x++)
    {
        array[x] = (rand() % 2);
    }
    for (int x = 128; x <= 1000000; x++)
    {
        array[x] = array[x - 1] ^ array[x - 127];

        if (array[x] == 0)
        {
            count0++;
        }
        if (array[x] == 1)
        {
            count1++;
        }

        int random = rand() % 2;

        if (random == 0)
        {
            rand0++;
        }
        if (random == 1)
        {
            rand1++;
        }
    }
    printf("%d %d\n", count0, count1);
    printf("%d %d\n", rand0, rand1);
    return 0;
}
```

OUTPUT:

	Number of 0s	Number of 1s
0s and 1s in x128 , · · , xN for N = 1e6	499530	500343
0s and 1s using rand() % 2 approach	500475	499398

PART 5(b):

First of all, we randomly generate bits for 1<=i<=127

Then, we iterate from 128<=i<=1000000

The possible cases are listed below:

x[i-1]	x[i-127]	x[i]
0	0	0
0	1	1
1	0	1
1	1	0

Picking the first and the last case from the truth table of an XOR gate as mentioned above, as we have to compute: P (xi = 0/x(i−1) = 0) and P (xi = 0/x(i−1) = 1)

CODE:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main()
{
    int count0 = 0, count1 = 0, array[1000000], array0 = 0, array1 = 0;
    srand(time(NULL));

    for (int i = 1; i <= 127; i++)
    {
        array[i] = (rand() % 2);
    }
    for (int i = 128; i <= 1000000; i++)
    {
        array[i] = array[i - 1] ^ array[i - 127];
```

```
        if (array[i - 1] == 0)
        {
            count0++;
        }
        if (array[i - 1] == 1)
        {
            count1++;
        }
        if ((array[i] == 0) && (array[i - 1] == 0))
        {
            array0++;
        }
        if ((array[i] == 0) && (array[i - 1] == 1))
        {
            array1++;
        }
    }

    printf("P(xi = 0/x(i−1) = 0) = %lf\n", (double)array0 / (double)count0);
    printf("P(xi = 0/x(i−1) = 1) = %lf\n", (double)array1 / (double)count1);
    return 0;
}
```

<u>OUTPUT:</u>

we obtain the following output after running the code, 5 iterations are listed as follows:

P(xi = 0/xi−1 = 0)	P(xi = 0/xi−1 = 1)
0.499987	0.500010
0.502192	0.500024
0.501612	0.499989
0.500834	0.500008
0.500037	0.499986