Assignment 2 (M207)
Prosondip Sadhukhan
Roll No-1911119

Q 1. *Write a note on a* ***primality test (Solovay-Strassen test or Miller-Robin test)****.*

Ans.

Solovay-Strassen Test:-The Solovay–Strassen primality test, developed by Robert M. Solovay and Volker Strassen in 1977, is a probabilistic test to determine if a number is composite or probably prime. The idea behind the test was discovered by M. M. Artjuhov in 1967.

Euler proved that for any prime number p and any integer a,

$$a^{(p-1)/2} \equiv \left(\frac{a}{p}\right) \pmod{p}$$

where $\left(\frac{a}{p}\right)$ is the Legendre symbol. The Jacobi symbol is a generalisation of the Legendre symbol to $\left(\frac{a}{n}\right)$, where n can be any odd integer. The Jacobi symbol can be computed in time $O((\log n)^2)$ using Jacobi's generalization of law of quadratic reciprocity.

Given an odd number n we can contemplate whether or not the congruence

$$a^{(n-1)/2} \equiv \left(\frac{a}{n}\right) \pmod{n}$$

holds for various values of the "base" a, given that a is relatively prime to n. If n is prime then this congruence is true for all a. So if we pick values of a at random and test the congruence, then as soon as we find an a which doesn't fit the congruence we know that n is not prime (but this does not tell us a nontrivial factorization of n). This base a is called an Euler witness for n; it is a witness for the compositeness of n. The base a is called an Euler liar for n if the congruence is true while n is composite.

For every composite odd n, at least half of all bases

$a \in (\mathbb{Z}/n\mathbb{Z})^*$

are (Euler) witnesses as the set of Euler liars is a proper subgroup of $(\mathbb{Z}/n\mathbb{Z})^*$. For example, for $n = 65$, the set of Euler liars has order 8 and $= \{1, 8, 14, 18, 47, 51, 57, 64\}$, and $(\mathbb{Z}/n\mathbb{Z})^*$ has order 48.

This contrasts with the Fermat primality test, for which the proportion of witnesses may be much smaller. Therefore, there are no (odd) composite n without many witnesses, unlike the case of Carmichael numbers for Fermat's test.

Solovay-Strassen Algorithm:-

Algorithm 5.6: SOLOVAY-STRASSEN(n)

choose a random integer a such that $1 \leq a \leq n-1$
$x \leftarrow \left(\frac{a}{n}\right)$
if $x = 0$
  then return ("n is composite")
$y \leftarrow a^{(n-1)/2} \pmod{n}$
if $x \equiv y \pmod{n}$
  then return ("n is prime")
  else return ("n is composite")

.

Example:- Suppose we wish to determine if n = 221 is prime. We write $(n-1)/2 = 110$.
We randomly select an a (greater than 1 and smaller than n): 47. Using an efficient method for raising a number to a power (mod n) such as binary exponentiation, we compute:
$a(n-1)/2(modn) = 47110(mod221) = -1(mod221)$
$(\frac{a}{n})(modn) = (\frac{47}{221})(mod221) = -1(mod221)$.
This gives that, either 221 is prime, or 47 is an Euler liar for 221.
We try another random a, this time choosing a = 2:
a(n-1)/2 (mod n)=2110(mod 221)=30 (mod 221)
$(\frac{a}{n})(modn) = (\frac{2}{221})(mod221) = -1(mod221)$. Hence 2 is an Euler witness for the compositeness of 221, and 47 was in fact an Euler liar. Note that this tells us nothing about the prime factors of 221, which are actually 13 and 17.

Accuracy of the Test:-It is possible for the algorithm to return an incorrect answer. If the input n is indeed prime, then the output will always correctly be probably prime. However, if the input n is composite then it is possible for the output to be incorrectly probably prime. The number n is then called an Euler–Jacobi pseudoprime.
When n is odd and composite, at least half of all a with gcd(a,n) = 1 are Euler witnesses. We can prove this as follows: let a1, a2, ..., am be the Euler liars and a an Euler witness. Then, for i = 1,2,...,m:

$$(a \cdot a_i)^{(n-1)/2} = a^{(n-1)/2} \cdot a_i^{(n-1)/2} = a^{(n-1)/2} \cdot \left(\frac{a_i}{n}\right) \not\equiv \left(\frac{a}{n}\right)\left(\frac{a_i}{n}\right) \pmod{n}.$$

Because the following holds:

$$\left(\frac{a}{n}\right)\left(\frac{a_i}{n}\right) = \left(\frac{a \cdot a_i}{n}\right),$$

now we know that

$$(a \cdot a_i)^{(n-1)/2} \not\equiv \left(\frac{a \cdot a_i}{n}\right) \pmod{n}..$$

This gives that each a_i gives a number $a \cdot a_i$, which is also an Euler witness. So each Euler liar gives an Euler witness and so the number of Euler witnesses is larger or equal to the number of Euler liars. Therefore, when n is composite, at least half of all a with gcd(a,n) = 1 is an Euler witness.
Hence, the probability of failure is at most 2^{-k} (compare this with the probability of failure for the Miller-Rabin primality test, which is at most 4^{-k}).
The bound 1/2 on the error probability of a single round of the Solovay–Strassen test holds for any input n, but those numbers n for which the bound is (approximately) attained are extremely rare. On the average, the error probability of the algorithm is significantly smaller: it is less than

$$2^{-k} \exp\left(-(1+o(1))\frac{\log x \log\log\log x}{\log\log x}\right)$$

for k rounds of the test, applied to uniformly random $n \leq x$.The same bound also applies to the related problem of what is the conditional probability of n being composite for a random number $n \leq x$ which has been declared prime in k rounds of the test.
Accuracy of the test can be increased by using **Square and multiply method**.
We can use the bits of the exponent in left to right order. In practice, we would usually want the result modulo some modulus m. In that case, we would reduce each multiplication result (mod m) before proceeding. For simplicity, the modulus calculation is omitted here. This example shows how to compute b^{13} using left to right binary exponentiation. The exponent is 1101 in binary; there

are 4 bits, so there are 4 iterations.
Initialize the result to 1:
$r \leftarrow 1\,(= b^0)$.
Step 1) $r \leftarrow r^2\,(= b^0)$; bit 1 = 1, so compute $r \leftarrow r \cdot b\,(= b^1)$;
Step 2) $r \leftarrow r^2\,(= b^2)$; bit 2 = 1, so compute $r \leftarrow r \cdot b\,(= b^3)$;
Step 3) $r \leftarrow r^2\,(= b^6)$; bit 3 = 0, so we are done with this step;
Step 4) $r \leftarrow r^2\,(= b^{12})$; bit 4 = 1, so compute $r \leftarrow r \cdot b\,(= b^{13})$

Algorithm of Square and Multiply Method:-

Algorithm 5.5: SQUARE-AND-MULTIPLY(x, c, n)

$z \leftarrow 1$
for $i \leftarrow \ell - 1$ **downto** 0
$$\textbf{do} \begin{cases} z \leftarrow z^2 \bmod n \\ \textbf{if } c_i = 1 \\ \quad \textbf{then } z \leftarrow (z \times x) \bmod n \end{cases}$$
return (z)

.

We can calculate jacobi symbol using this properties:-

1.If n is (an odd) prime, then the Jacobi symbol $\left(\frac{a}{n}\right)$ is equal to (and written the same as) the corresponding Legendre symbol.
2. If $a = b(mod n)$, then $\left(\frac{a}{n}\right) = \left(\frac{b}{n}\right) = \left(\frac{a \pm m \cdot n}{n}\right)$.
3. $\left(\frac{a}{n}\right) = \begin{cases} 0 & \text{if } \gcd(a,n) \neq 1, \\ \pm 1 & \text{if } \gcd(a,n) = 1. \end{cases}$
If either the top or bottom argument is fixed, the Jacobi symbol is a completely multiplicative function in the remaining argument:
4. $\left(\frac{ab}{n}\right) = \left(\frac{a}{n}\right)\left(\frac{b}{n}\right), \quad \text{so} \quad \left(\frac{a^2}{n}\right) = \left(\frac{a}{n}\right)^2 = 1 \text{ or } 0.$
5. $\left(\frac{a}{mn}\right) = \left(\frac{a}{m}\right)\left(\frac{a}{n}\right), \quad \text{so} \quad \left(\frac{a}{n^2}\right) = \left(\frac{a}{n}\right)^2 = 1 \text{ or } 0.$

The law of quadratic reciprocity: if m and n are odd positive coprime integers, then

6. $\left(\frac{m}{n}\right)\left(\frac{n}{m}\right) = (-1)^{\frac{m-1}{2}.\frac{n-1}{2}} = \begin{cases} 1 & \text{if } n \equiv 1 \pmod 4 \text{ or } m \equiv 1 \pmod 4, \\ -1 & \text{if } n \equiv m \equiv 3 \pmod 4 \end{cases}$

and its supplements

7. $\left(\frac{-1}{n}\right) = (-1)^{\frac{n-1}{2}} = \begin{cases} 1 & \text{if } n \equiv 1 \pmod 4, \\ -1 & \text{if } n \equiv 3 \pmod 4, \end{cases}$
8. $\left(\frac{2}{n}\right) = (-1)^{\frac{n^2-1}{8}} = \begin{cases} 1 & \text{if } n \equiv 1,7 \pmod 8, \\ -1 & \text{if } n \equiv 3,5 \pmod 8. \end{cases}$

Combining property 4 and 8 gives:

9. $$\left(\frac{2a}{n}\right)=\left(\frac{2}{n}\right)\left(\frac{a}{n}\right)=\begin{cases}\left(\frac{a}{n}\right) & \text{if } n\equiv 1,7 \pmod 8,\\ -\left(\frac{a}{n}\right) & \text{if } n\equiv 3,5 \pmod 8.\end{cases}$$

Algorithm of Calculating Jacobi symbol:-

Algorithm 2.1 QuadraticBinaryJacobi

Input: $a, b \in \mathbb{N}$ with $\nu(a) = 0 < \nu(b)$

Output: Jacobi symbol $(b|a)$

$s \leftarrow 0, \quad j \leftarrow \nu(b)$

while $2^j a \neq b$ **do**

  $b' \leftarrow b/2^j$

  $s \leftarrow (s + j(a^2 - 1)/8) \bmod 2$

  $(q, r) \leftarrow \mathrm{BinaryDividePos}(a, b)$

  if $(j, q) = (1, 3)$ **then**

    $d \leftarrow a - b'$

    $m \leftarrow \nu(d) \operatorname{div} 2$

    $c \leftarrow (d - (-1)^m d/4^m)/5$

    $s \leftarrow (s + m(a - 1)/2) \bmod 2$

    $(a, b) \leftarrow (a - 4c, b + 2c)$ ▷ harmless iteration

  else

    $s \leftarrow (s + (a - 1)(b' - 1)/4) \bmod 2$

    $(a, b) \leftarrow (b', r/2^j)$ ▷ good or bad iteration

  $s \leftarrow (s + j(a^2 - 1)/8) \bmod 2, \quad j \leftarrow \nu(b)$

if $a = 1$ **then** return $(-1)^s$ **else** return 0

.

Q 2. *Implement in C/C++/Python*

Ans.

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#include <stdlib.h>
#include<time.h>
int Square_and_multiply(int x, int c, int n)
{
	int z=1,e[100],k=0,i;
	while (c>0)
	{
		e[k]=(c%2);
		c=(c/2);
		k++;
	}
	for(i=k-1;i>-1;i--)
	{
		z=((z*z)%n);
		if (e[i]==1)
		{
			z=((z*x)%n);
		}
	}
	return z;
}
int gcd(int k, int r)
{
if(r==0)
	{
		return k;
	}
else
	{
		return gcd(r,(k%r));
	}
}
int Jacobi(int a, int n, int b)
{
	int p=0;
	if(gcd(a,n)>1)
	{
		return 0;
	}
	else
	{
		if(a%2==1 && a!=1)
		{
			if(a<n)
			{
				if((n%4)==3 && (a%4)==3)
				{
					b=b*(-1);
				}
				return Jacobi(n,a,b);
			}
			else
			{
				return Jacobi((a%n),n,b);
			}
```

```c
			}
			else
			{
				while(a%2==0)
				{
					a=a/2;
					p++;
				}
				if((p%2==1 && n%8==3) || (p%2==1 && n%8==5))
				{
					b=b*(-1);
				}
				if(a==1)
				{
					return b;
				}
				else
				{
					return Jacobi(a,n,b);
				}
			}
		}
		if(a==1)
			{
				return b;
			}
}
int main()
{
	int a,n,b,c,d,e=0;
	O:printf("\n Enter a number to check it is prime or not using Solovay-Strassen method:-");
	scanf("%d",&n);
	if(n<1)
	{
		printf("Enter valid Natural number.");
		goto O;
	}
	if (n==1)
	{
		printf("\n %d is not a composite or not a prime number.",n);
		return 0;
	}
	for(b=0;b<500;b++)
	{
		a=rand() % (n-1)+1;
		c=((Jacobi(a,n,1)+n)%n);
		if (c==0)
		{
			printf("\n %d is composite.",n);
			e=1;
			break;
		}
		d=Square_and_multiply(a,((n-1)/2),n);
		if(d!=c)
		{
			printf("\n %d is composite.",n);
			e=1;
			break;
		}
```

```

    }
    if(e==0)
    {
        printf("\n %d is prime.",n);
    }
    return 0;
}
```