IA5:7 Primarity Tests.

To solve Programming problems which includes the concept of prime number and its related properties.

a. sieve of eratosthenes with example.
b. fermats primality testing with example.
c. Miller - Robin primality Testing with example.

(a) Sieve of eratosthenes with example:-

The sieve of eratosthenes is an efficient algorithm to find all prime number to a given number.

steps:

1. Create a list of number from 2 to n
2. start with the first prime (2).
3. eliminate all multiples of 2.
4. Repeat until you reach $\sqrt{n}$
5. The remaining unmarked number are all prime.

example in C:

```
#include <stdio.h>
#include <stdbool.h>
void sieve of eratosthenes (int n){
  bool prime[n+1]
  for (int i = 0; i<n; i++)
```

```
prime[i] = true;
prime[0] = prime[1] = false;
for(int p = 2; p*p <= n; p++)
    prime[i] = false;
}
printf("prime number up to %d are: \n", n);
for(int i = 2; i <= n; i++){
  if(prime[i])
    printf("%d ", i);
  }
  printf("\n");
}
int main(){
  int n;
  printf("enter the limit: ");
  scanf("%d", &n);
  sieve of eratosthenes(n);
  return 0;
}
```

B. fermat's primality testing with example:

fermat is a little theorem:

If p is prime and a is any integer such that the If p is prime and a is any integer

$a^{p-1} \equiv 1 \pmod{p}$ where $a^{(p-2)} \equiv z \pmod{p}$ or $a \cdot z \equiv 1 \pmod{p}$

Idea:

- pick a random number a in range [2, p-2]
- compute $a^{p-1} \bmod p$ at [p-1] \mod -1 mod p
- If result $\neq 1 \rightarrow p$ is composite

Note: It is a probabilistic test. carmichael number may pass even though they are not prime.

example in C:

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

long long power(long long a, long long b, long long mod){
    long long result = 1;
    a = a % mod;
    while (b > 0){
        if(b & 1)
            result = (result * a) % mod;
        b = b >> 1;
        a = (a * a) % mod;
    }
    return result;
}

int isPrimeFermat(int n, int k){
    if(n <= 1 || n == 4) return 0;
```

```
if (n <= 3) return 1;
}
int main() {
  srand(time(0));
  int n,k;
  printf("enter number to test: ");
  scanf("%d", &n);
  printf("enter number of iterations: ");
  scanf("%d",&k);
  if (isPrimeFermat(n,k))
    printf("%d is probably prime.\n", n);
  else
    printf("%d is composite.\n", n);
  return 0;
}
```

Concept: Miller-Rabin Primality test.

If is a probistic test but much stronger than fermat's test

Ideas:-

1. write $n-1 = 2^s d$ $n-1 = 2^s \cdot d$, i.e. $n-1 = 2^s - d$, where d is odd
2. pick a random number a in $[2, n-2]$
3. compare $x = a^d \bmod n$ $x = a^d \bmod n = a^d \bmod n$
4. otherwise, square x repeatedly
- If you get error $n-1$ $n-1$ $n-1$, then continue

- If you never get n-1 n-1 n-1 then n is composite.
5. Repeat the test multiple times with different random a.
6. If it passes all in is probably prime.

example c program:

```
#include <stdio.h>
#include<stdlib.h>
#include<time.h>
long long long power(long long a, long long b, long long mod)
{
long long result=1;
a=a%mod;
while(b>0){
 if(b&1)
  result=(result*a)%mod;
  b=b>>1;
  a=(a*a)%mod;
 }
 return result;
}
int millerTest(long long d, long long n){
  long long a=2+rand()%(n-4);
  long long x=power(a,d,n);
  if(x==1||x==n-1)
   return 1;
```

```
while(d!=n-1){
  x=(x*x)%n;
  d=2;
if(x==1) return 0;
if(x==n-1) return 1;
}
return 0;
}
int is prime Miller Robin (long long n, int k){
  if n<=1||n==4) return 0;
  if(n<=3) return 1;
}
int main(){
  srand(time(0));
long long n;
int k;
printf("enter number to test:");
scanf("%d", &k);
if(is prime Miller Robin (n,k))
   printf("%lld is probably prime\n", n);
else
  printf("%lld is composite\n", n);
return 0;
}
```

output:

```
enter number to test is:97
enter number to iteration: 5
97 is probably prime.
```

VEL TECH - CSE	
EX NO.	7
PERFORMANCE (5)	5
RESULT AND ANALYSIS (3)	3
VIVA VOCE (3)	
RECORD (4)	
TOTAL (15)	8
SIGN WITH DATE	