3. Theta Notation, Θ

The notation $\Theta(n)$ is the formal way to express both the lower bound and the upper bound of an algorithm's running time. It is represented as

$$\Theta(f(n)) = \{ g(n) \text{ if and only if } g(n) = O(f(n)) \text{ and } g(n) = \Omega(f(n)) \text{ for all } n > n_0 \}$$

13-2-26

Short Notes on Little Omega —

The little omega (ω) notation is a method of expressing the lower bound on the growth rate of an algorithm's running time which may or may not be asymptotically tight therefore little omega (ω) is also called a loose lower bound. We use little omega notation (ω) to denote lower bound that is asymptotically not tight. Little omega can formally be defined as —

Given functions $f(n)$ and $g(n)$, we say that $f(n)$ is a little omega of $g(n)$ if there are positive constants c and n_0 such that $f(n) > c\,g(n)$ for all n, $n \geq n_0$

that is, f has a higher growth rate than g - so little omega (ω) is to mean "tight lower bound".

Ex:- Prove that - $4n + 6 = \omega(1)$

The Little omega (ω) running time can be proved by applying the limit formula as given below

If $\lim_{n \to \infty} \frac{f(n)}{g(n)} = \infty$, then function $f(n)$ is $\omega(g(n))$.

Here, we have $f(n) = 4n + 6$

$g(n) = 1$

So, $\lim_{n \to \infty} \frac{f(n)}{g(n)} = \lim_{n \to \infty} \frac{4n+6}{1}$

$= \lim_{n \to \infty} \left(\frac{4n}{1} + \frac{6}{1}\right) = \lim_{n \to \infty} (4n+6)$

Therefore, $f(n)$ is $\omega(g(n))$

So, $4n + 6 = \omega(1)$.

Q. Write a program in C to (calculate) find product of two matrices.

Ans→

```
#include<stdio.h>
#include<conio.h>
void main(){ int i, j, K, r1, c1, r2, c2;
  int a[10][10], b[10][10], c[10][10];
printf("\nEnter the number of rows and columns of first matrix");
  scanf("%d%d", &r1, &c1);
printf("\nEnter the no of rows & columns of 2nd Matrix:");
  scanf("%d %d", &r2, &c2);
  if (c1==r2)
  { printf("\n Enter numbers to matrix 1\n");
  for(i=0; i<r1; i++)
   { for(j=0; j<c1; j++)
     { scanf("%d", &a[i][j]);
     }
  }
```

```
printf("\n Enter Numbers to matrix 2 :");
for (i=0; i<r2; i++)
{
    for (j=0; j<c2; j++)
    {
        scanf ("%d", &b[i][j]);
    }
}
for (i=0; i<r1; i++)
{
    for (j=0; j<c2; j++)
    {
        c[i][j] = 0;
        for (k=0; k<r1; k++)
        {
            c[i][j] = c[i][j] + a[i][k] * b[k][j];
        }
    }
}
printf("\n The First Matrix elements :\n");
for (j=0; i<r1; i++)
{
    for (j=0; j<c1; j++)
    {
        printf ("%d ", a[i][j]);
    }
    printf("\n");
}
printf("\n The Second Matrix elements :\n");
for (i=0; i<r2; i++)
{
    for (j=0; j<c2; j++)
    {
        printf ("%d ", b[i][j]);
    }
    printf("\n");
}
```

```
Printf("\n The Product of both Matrices: \n");
for(i=0; i<r1; i++)
 {
   for(j=0; j<c2; j++)
    {
      printf(" %d ", c[i][j]);
    }
  printf("\n");
   }

} //close of validation. for no of column = No of row
  else
      printf("Matrix multiplication not possible");
getch();
} //close of main
```