# PRACTICAL – 1

**1.1 AIM: Factorial (Iterative and Recursive)**

**PROGRAM CODE**: **Using Iterative Method:**

#include <iostream> using namespace std; int main()

{

cout << "Using Iterative Method\n\n"; long int n, fact = 1, counter = 0;

cout << "Enter number:"; cin >> n;

for (int i = 1; i <= n; i++)

{

fact = fact \* i; counter++;

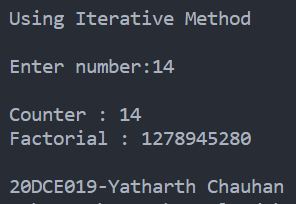
}

cout << "\nCounter : " << counter; cout << "\nFactorial : " << fact;

cout << "\n\n20DCE019-Yatharth Chauhan";

}

**OUTPUT**:



**ANALYSIS TABLE**:

|  |  |
| --- | --- |
| Input | Counter |
| 12 | 12 |
| 23 | 23 |
| 5 | 5 |
| 8 | 8 |

**GRAPH**:

**PROGRAM CODE**: **Using Recursive Method:**

#include <iostream> using namespace std; int counter = 0;

long int factfunc(long int n)

{

if (n == 0) return 1;

else

{

counter++;

return factfunc(n - 1) \* n;

}

}

int main()

{

cout << "Using Recursion Method:\n\n"; long int n, fact = 1;

cout << "Enter number: "; cin >> n;

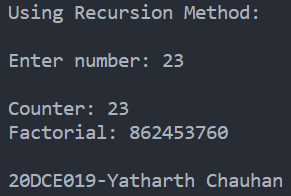
fact = factfunc(n);

cout << "\nCounter: " << counter; cout << "\nFactorial: " << fact;

cout << "\n\n20DCE019-Yatharth Chauhan";

}

**OUTPUT**:



**ANALYSIS TABLE**:

|  |  |
| --- | --- |
| Input | Counter |
| 1 | 1 |
| 7 | 7 |
| 13 | 13 |
| 17 | 17 |
| 20 | 20 |

**GRAPH**:

**CONCLUSION:**

In this practical I implemented the factorial of given number by both iterative and recursi

**1.2 AIM: Fibonacci (Iterative and Recursive)**

**PROGRAM CODE**: **Using Iterative Method**

#include <iostream> using namespace std; int count = 0;

void fib(int N)

{

int first = 0, second = 1, next; for (int i = 0; i < N; i++)

{

if (i <= 1)

{

count++; next = i;

}

else

{

next = first + second; first = second; second = next; count++;

}

cout << next << ' ';

}

}

int main()

{

int Number, i = 0;

cout << "Enter number for fibonacci terms: "; cin >> Number;

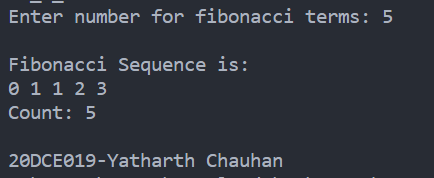
cout << "\nFibonacci Sequence is: \n"; fib(Number);

cout << "\nCount: " << count << endl;

cout << "\n20DCE019-Yatharth Chauhan"; return 0;

}

**OUTPUT:**



**ANALYSIS TABLE:**

|  |  |
| --- | --- |
| INPUT | COUNTER |
| 1 | 1 |
| 3 | 3 |
| 5 | 5 |

**GRAPH:**

**PROGRAM CODE**: **Using Recursive Method**

#include <iostream> using namespace std; int count = 0;

int fibonacci(int n)

{

if ((n == 1) || (n == 0))

{

count++; return (n);

}

else

{

count++;

return (fibonacci(n - 1) + fibonacci(n - 2));

}

}

int main()

{

int n, i = 0;

cout << "Input the number of terms for Fibonacci Series: "; cin >> n;

cout << "\nFibonacci Series: ";

while (i < n)

{

cout << " " << fibonacci(i); i++;

}

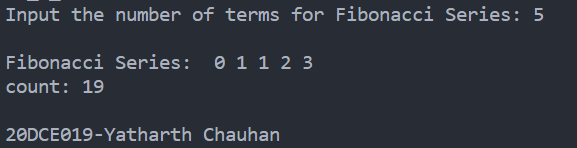
cout << "\ncount: " << count << endl;

cout << "\n20DCE019-Yatharth Chauhan";

return 0;

}

**OUTPUT:**



**CONCLUSION**:

As for both the methods, the order of the iterative method is O(n) whereas using the recursive method, the order is of O(n^2).

**1.3 AIM: Matrix Addition and Matrix Multiplication (Iterative)**

**PROGRAM CODE**:

#include <iostream> using namespace std;

int count = 0, cnt = 0; int main()

{

int a[10][10], b[10][10], mul[10][10], sum[100][100], r, c, i, j, k;

cout << "Enter the number of row: "; cin >> r;

cout << "Enter the number of column: "; cin >> c;

cout << "\nEnter the elements of first matrix: \n"; for (i = 0; i < r; i++)

{

for (j = 0; j < c; j++)

{

cin >> a[i][j];

}

}

cout << "\nEnter the elements of second matrix: \n";

for (i = 0; i < r; i++)

{

for (j = 0; j < c; j++)

{

cin >> b[i][j];

}

}

for (i = 0; i < r; i++)

{

for (j = 0; j < c; j++)

{

mul[i][j] = 0;

for (k = 0; k < c; k++)

{

cnt++;

mul[i][j] += a[i][k] \* b[k][j];

}

}

}

// adding two matrix for(i = 0; i < r; ++i)

{

for (j = 0; j < c; ++j)

{

count++;

sum[i][j] = a[i][j] + b[i][j];

}

}

// for printing result

cout << "\nMatrix Multiplication :" << endl; for (i = 0; i < r; i++)

{

for (j = 0; j < c; j++)

{

cout << mul[i][j] << " ";

}

cout << "\n";

}

// Displaying the resultant sum matrix. cout << "\nMatrix Addition :" << endl; for (i = 0; i < r; ++i)

for (j = 0; j < c; ++j)

{

cout << sum[i][j] << " "; if (j == c - 1)

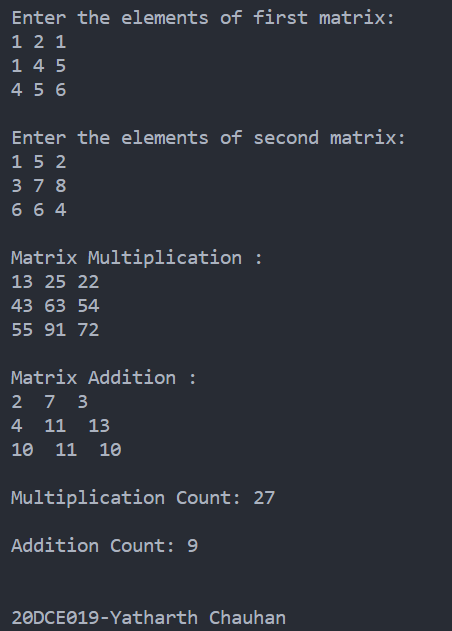
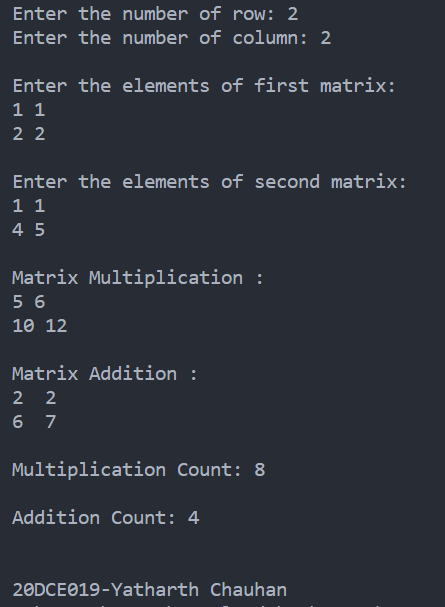
cout << endl;

}

cout << "\nMultiplication Count: " << cnt << endl; cout << "\nAddition Count: " << count << endl; cout << "\n\n20DCE019-Yatharth Chauhan"; return 0;

}

**OUTPUT**:



**ANALYSIS TABLE**:

|  |  |  |
| --- | --- | --- |
| Input | Addition Count | Multiplication Count |
| 2 | 4 | 8 |
| 3 | 9 | 27 |
| 5 | 25 | 125 |
| 7 | 49 | 343 |

**GRAPH**:

**CONCLUSION:**

Here we concluded by implementing and analyzing the code of matrix addition and multiplication, the time complexity of matrix addition is O(n^2) and of matrix multiplication is O(n^3) by analyzing the graph of matrix addition is linear whereas of matrix multiplication is slightly curved in beginning and then linear.

**1.4 AIM: Find a subset of a given set S = {s1,s2,. , sn} of n positive integers**

**whose sum is equal to a given positive integer d. For example, if S= {1, 2, 5, 6, 8} and d = 9 there are two solutions {1,2,6} and {1,8}. A suitable message is to be displayed if the given problem instance doesn't have a solution.**

**PROGRAM CODE**:

#include <iostream> using namespace std; int counter = 0;

bool isSubsetSum(int set[], int n, int sum)

{

counter++;

if (sum == 0) return true;

if (n == 0 && sum != 0) return false;

if (set[n - 1] > sum)

return isSubsetSum(set, n - 1, sum);

return isSubsetSum(set, n - 1, sum) || isSubsetSum(set, n - 1, sum - set[n - 1]);

}

int main()

{

int set[] = {1, 1, 2, 6, 8, 7, 4};

int d = 12;

int n = sizeof(set) / sizeof(set[0]); cout << "no of Inputs: " << n << endl; cout << "The Sum is: " << d;

if (isSubsetSum(set, n, d) == true)

cout << "\nFound the subset with given sum"; else

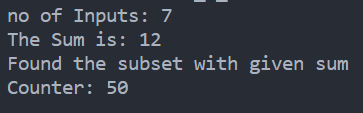
cout << "No subset with given sum"; cout << "\nCounter: " << counter << endl;

cout << "\n\n\n20DCE019-Yatharth Chauhan";

return 0;

}

**OUTPUT**:



**ANALYSIS TABLE**:

|  |  |
| --- | --- |
| Input | Count |
| 3 | 15 |
| 5 | 27 |
| 8 | 47 |
| 10 | 68 |
| 14 | 128 |

**GRAPH**:

**CONCLUSION:**

Here we concluded by implementing and analyzing the code of subset sum the time complexity of subset sum is O(2^n) and by analyzing the graph is non-linear.