```cpp
#include <iostream>
#include <ctime>
#include <iomanip>

using namespace std;

int const FibonacciMaximumNumber = 12;

int FibonacciRecursive(int n); // recursive algorithm
int FibonacciModifiedRecursive(int n); // algorithm modified recursive
int FibonacciNonRecursive(int n); // algorithm non recursive


int main() {
    clock_t TimeRequired; // time needed for execution
    int FibonacciNumbers[FibonacciMaximumNumber] = { 1,5,10,15,20,25,30,35,40,45,50,55 };
```

```
    float FibonacciTimesRecursive[FibonacciMaximumNumber]; // Array for storing the Fibonacci time obtained through recursion
    int FibonacciValuesRecursive[FibonacciMaximumNumber]; // Array for storing the Fibonacci values obtained through recursion

    float FibonacciModifiedRecursiveTimes[FibonacciMaximumNumber]; // Array for storing the Fibonacci time obtained through modified recursion
    int FibonacciModifiedRecursiveValues[FibonacciMaximumNumber]; // Array for storing the Fibonacci time obtained through modified recursion

    float FibonacciTimesNonRecursive[FibonacciMaximumNumber]; // Array for storing the Fibonacci time obtained through non recursion
    int FibonacciValuesNonRecursive[FibonacciMaximumNumber]; // Array for storing the Fibonacci values obtained through non recursion
```

```
    // head of the table
    cout << left << setw(20) << "Numbers"
         << setw(20) << "Recursion (s)"
         << setw(30) << "Modified Recursive (s)"
         << setw(24) << "Non Recursive (s)"
         << setw(23) << "Fibonacci Values"
         << endl;
    cout <<
"=============================================================
=============================================================
==================" << endl;
    // ALGORITHMS
    // Algorithm that employs recursion
    for (int i = 0; i < FibonacciMaximumNumber;
i++) {
        TimeRequired = clock();  // timer begin
        FibonacciValuesRecursive[i] =
FibonacciRecursive(FibonacciNumbers[i]); // run
algorithm recursive
        TimeRequired = clock() -
TimeRequired; // timer finish
```

```
        FibonacciTimesRecursive[i] =
TimeRequired; // Save time duration in seconds
    }

    // Algorithm that employs modified recursive
    for (int i = 0; i < FibonacciMaximumNumber;
i++) {
        TimeRequired = clock(); // timer begin
        FibonacciModifiedRecursiveValues[i] =
FibonacciModifiedRecursive(FibonacciNumbers[i]);
// run algorithm modified recursive
        TimeRequired = clock() -
TimeRequired; // timer finish
        FibonacciModifiedRecursiveTimes[i] =
TimeRequired; // // Save time duration in
seconds
    }

    // Algorithm that employs non recursive
    for (int i = 0; i < FibonacciMaximumNumber;
i++) {
        TimeRequired = clock(); // timer begin
```

```
        FibonacciValuesNonRecursive[i] =
FibonacciNonRecursive(FibonacciNumbers[i]); //
run algorithm non recursive
        TimeRequired = clock() -
TimeRequired; // timer finish
        FibonacciTimesNonRecursive[i] =
TimeRequired; // Save time duration in seconds
    }
```

```
    // table body
    for (int i = 0; i < FibonacciMaximumNumber;
i++) {
        cout << left << setw(20) <<
FibonacciNumbers[i] << setw(20)
             <<
(float)FibonacciTimesRecursive[i] /
CLOCKS_PER_SEC
             << setw(30) <<
(float)FibonacciModifiedRecursiveValues[i] /
CLOCKS_PER_SEC
             << setw(24) <<
(float)FibonacciTimesNonRecursive[i] /
CLOCKS_PER_SEC
```

```
            << setw(23) <<
FibonacciValuesRecursive[i]
            << endl;
    }
}
```

```
// recursive algorithm
int FibonacciRecursive(int n) {
    if (n == 0 || n == 1) return(n);
    else return(FibonacciRecursive(n - 1) +
FibonacciRecursive(n - 2));
}
```

```
int fi[10000];
// algorythm modified recursive
int FibonacciModifiedRecursive(int n) {
    if (n == 0 || n == 1) return n;
    if (fi[n] != -1) return fi[n];
    fi[n] = FibonacciModifiedRecursive(n - 1) +
FibonacciModifiedRecursive(n - 2);
    return fi[n];
```

```
}
```

```
// algorithm non recursive
int FibonacciNonRecursive(int n) {
    if (n <= 1) return n;

    int fibonacci = 1;
    int fibonacciPrevious = 1;
    for (int i = 2; i < n; ++i)
    {
        int temp = fibonacci;
        fibonacci += fibonacciPrevious;
        fibonacciPrevious = temp;
    }
    return fibonacci;
}
```