COMP133 –COMPUTER AND PROGRAMMING

Recursion

Dr. Radi Jarrar
Department of Computer Science
Birzeit University

Recursion

- A function that calls itself or part of a cycle in the sequence of a function call.
- Recursion can be used as alternative to iterations (loops).
- Function calls cause overhead on the computation in means of time efficiency. However, recursion may provide a more natural solution to problems than iterations.

Recursion (2)

- Writing recursive algorithm

```
If this is a simple case
    solve it
Else
    redefine the problem using recursion
```

Recursion (3)

- A recursive algorithm should be established by verifying the following 2 properties:

1. The algorithm should have at least one base case.
2. Every recursive call gets closer to the base case in such a way that the base case will eventually be reached.

Recursion (5)

- Example: Write a C-program to find the sum of the first n natural numbers using recursion. Note: natural numbers are the positive integers.

Recursion (5)

```
#include<stdio.h>
int sum( int n){
      if(n == 0)
            return n;
      else
            return n + sum(n - 1);
}
int main()
      int num, add;
      printf("Enter a positive integer:\n);
      scanf("%d", &num);
      add = sum( num );
      printf("sum=%d", add);
      return 0;
}
```

Recursion (6)

- Visualise the recursive call:

sum(5) = 5 + sum (4)

Recursion (6)

- Visualise the recursive call:

sum(5) = 5 + sum (4)

= 5 + 4 + sum(3)

Recursion (6)

- Visualise the recursive call:

sum(5) = 5 + sum (4)

= 5 + 4 + sum(3)

= 5 + 4 + 3 + sum(2)

Recursion (6)

- Visualise the recursive call:

$$\begin{aligned} sum(5) &= 5 + sum(4) \\ &= 5 + 4 + sum(3) \\ &= 5 + 4 + 3 + sum(2) \\ &= 5 + 4 + 3 + 2 + sum(1) \end{aligned}$$

Recursion (6)

- Visualise the recursive call:

$$
\begin{aligned}
\text{sum}(5) &= 5 + \text{sum}(4) \\
&= 5 + 4 + \text{sum}(3) \\
&= 5 + 4 + 3 + \text{sum}(2) \\
&= 5 + 4 + 3 + 2 + \text{sum}(1) \\
&= 5 + 4 + 3 + 2 + 1 + \text{sum}(0)
\end{aligned}
$$

Recursion (6)

- Visualise the recursive call:

$$
\begin{aligned}
\text{sum}(5) &= 5 + \text{sum}(4) \\
&= 5 + 4 + \text{sum}(3) \\
&= 5 + 4 + 3 + \text{sum}(2) \\
&= 5 + 4 + 3 + 2 + \text{sum}(1) \\
&= 5 + 4 + 3 + 2 + 1 + \text{sum}(0) \\
&= 5 + 4 + 3 + 2 + 1 + 0
\end{aligned}
$$

Recursion (6)

- Visualise the recursive call:

$$
\begin{aligned}
\text{sum}(5) &= 5 + \text{sum}(4) \\
&= 5 + 4 + \text{sum}(3) \\
&= 5 + 4 + 3 + \text{sum}(2) \\
&= 5 + 4 + 3 + 2 + \text{sum}(1) \\
&= 5 + 4 + 3 + 2 + 1 + \text{sum}(0) \\
&= 5 + 4 + 3 + 2 + 1 + 0 \\
&= 15
\end{aligned}
$$

Recursion (7)

- Write a recursive implementation of the multiplication between two numbers.

Recursion (7)

- Write a recursive implementation of the multiplication between two numbers.

```
int multiply( int m, int n ){
     int answer;

     if(n == 1)
          answer = m;
     else
          answer = m + multiply(m, n - 1);

     return answer;
}
```

Recursion (8)

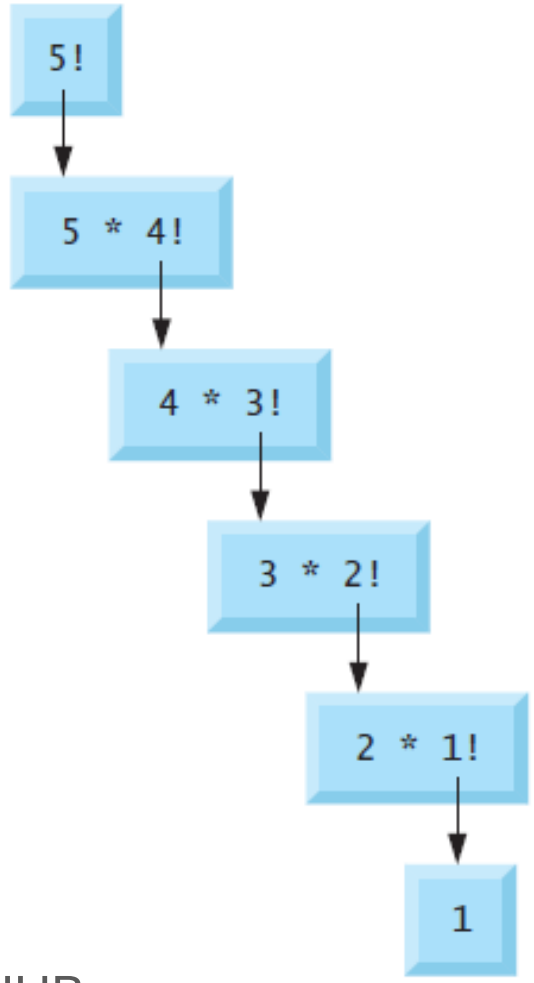
- Write a recursive implementation of the factorial.

Recursion (8)

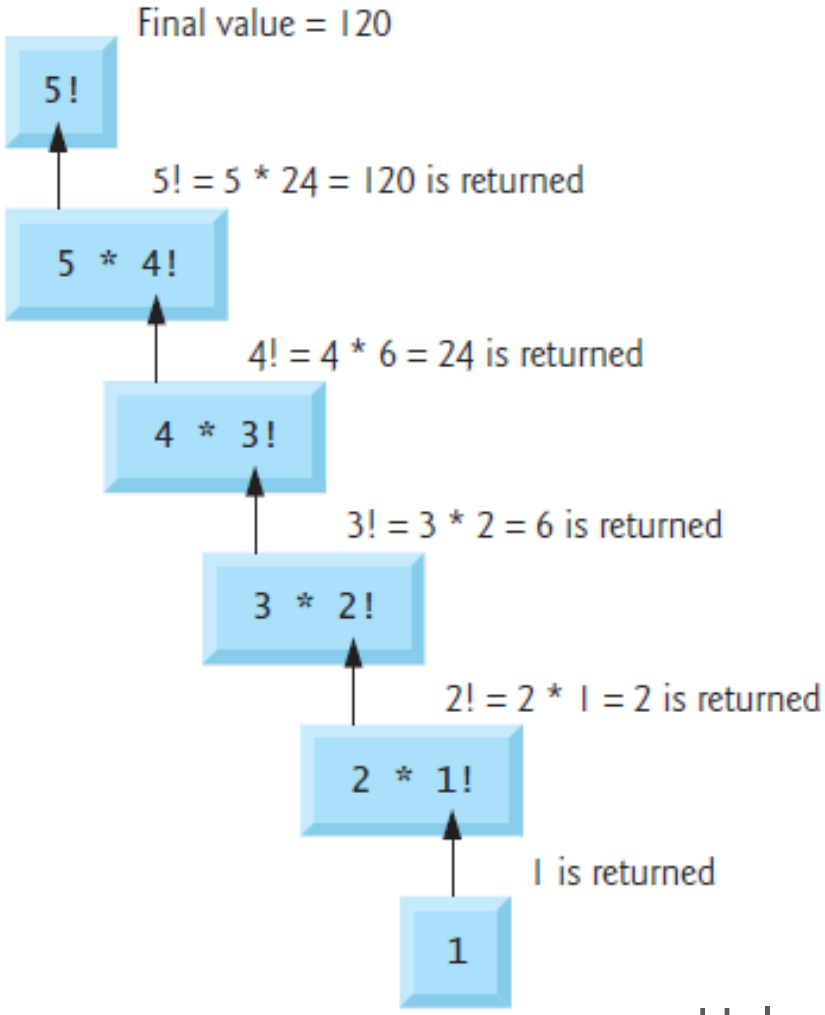
- Write a recursive implementation of the factorial.

```
int factorial( int n ){
     int answer;

     if(n == 0)
          answer = 1;
     else
          answer = n * factorial(n – 1);

     return answer;
}
```

Recursion (9)



(a) Sequence of recursive calls

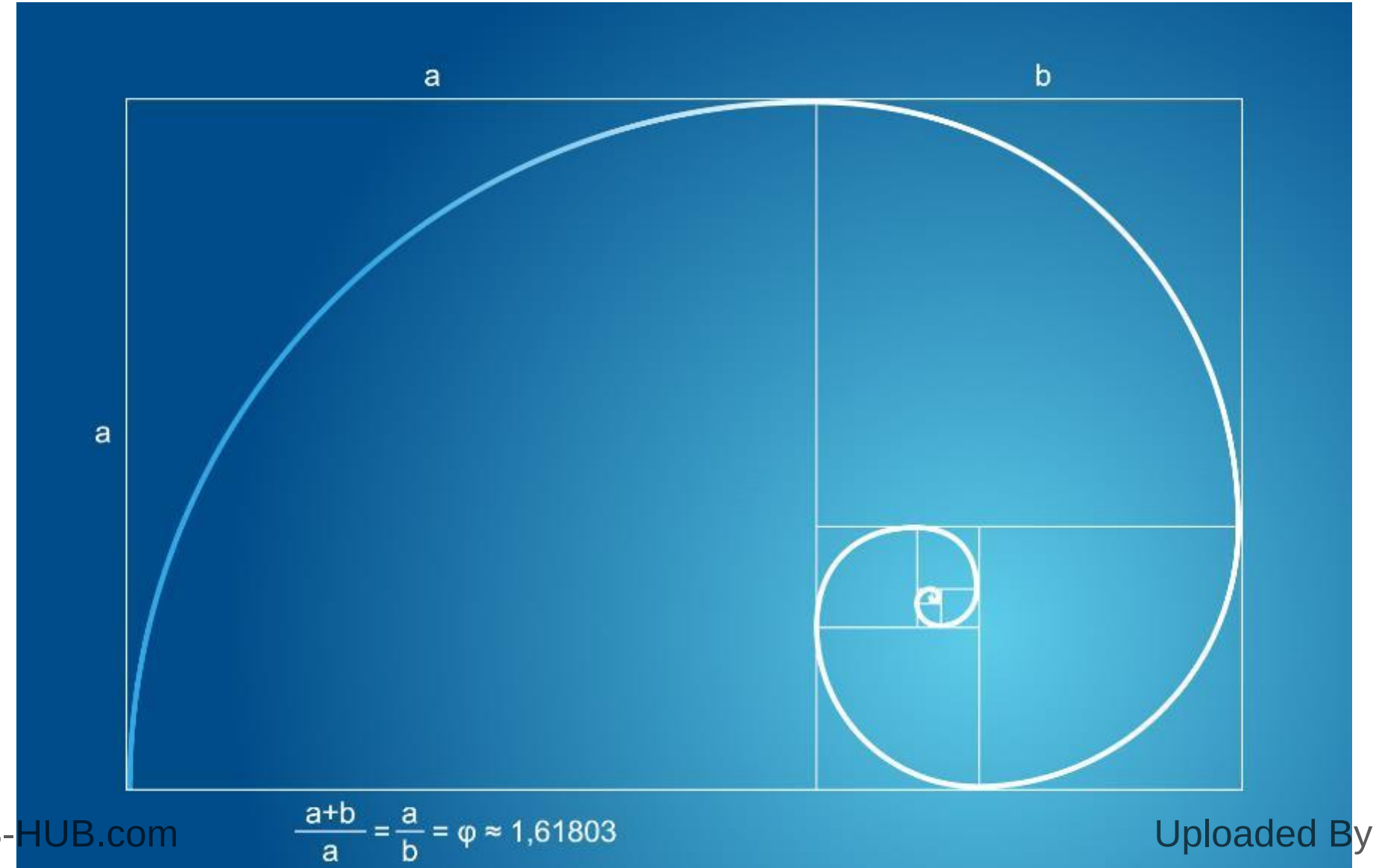


(b) Values returned from each recursive call

Fibonnacci series

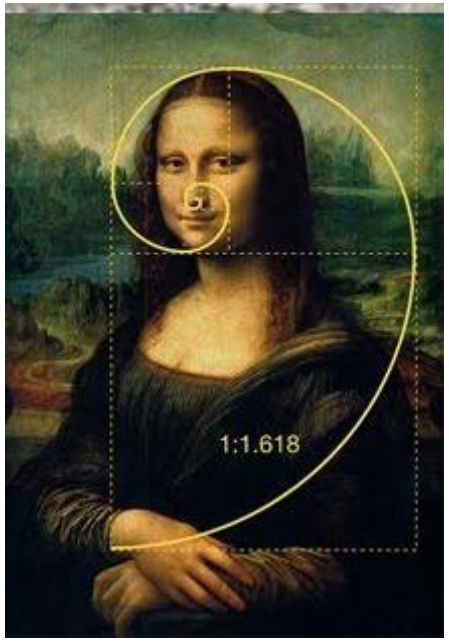
- The sequence is 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, …
- Each element equals the sum of its previous two consecutive elements.
- It was first developed to model the growth of a rabbit colony.
- Fib(0) = 0
- Fib(1) = 1
- Fib(n) = Fib(n - 2) + Fib(n - 1) for n>2

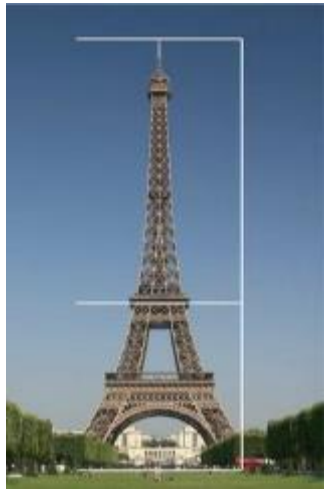
Applications to Fibonnacci – Architecture

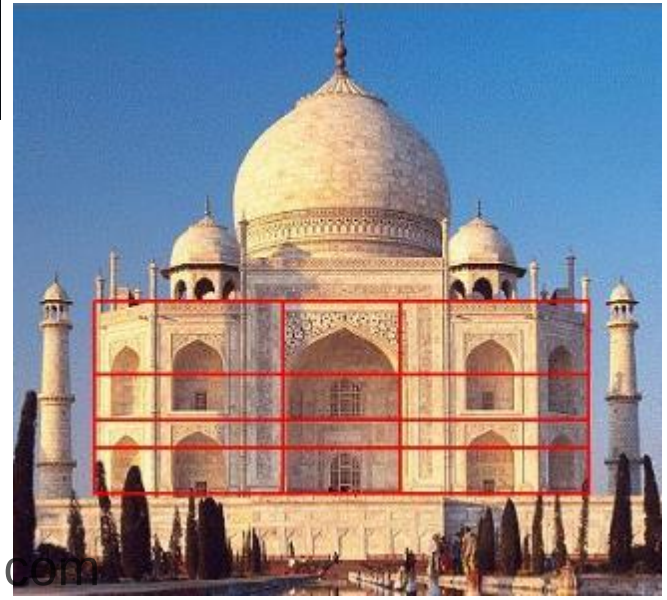


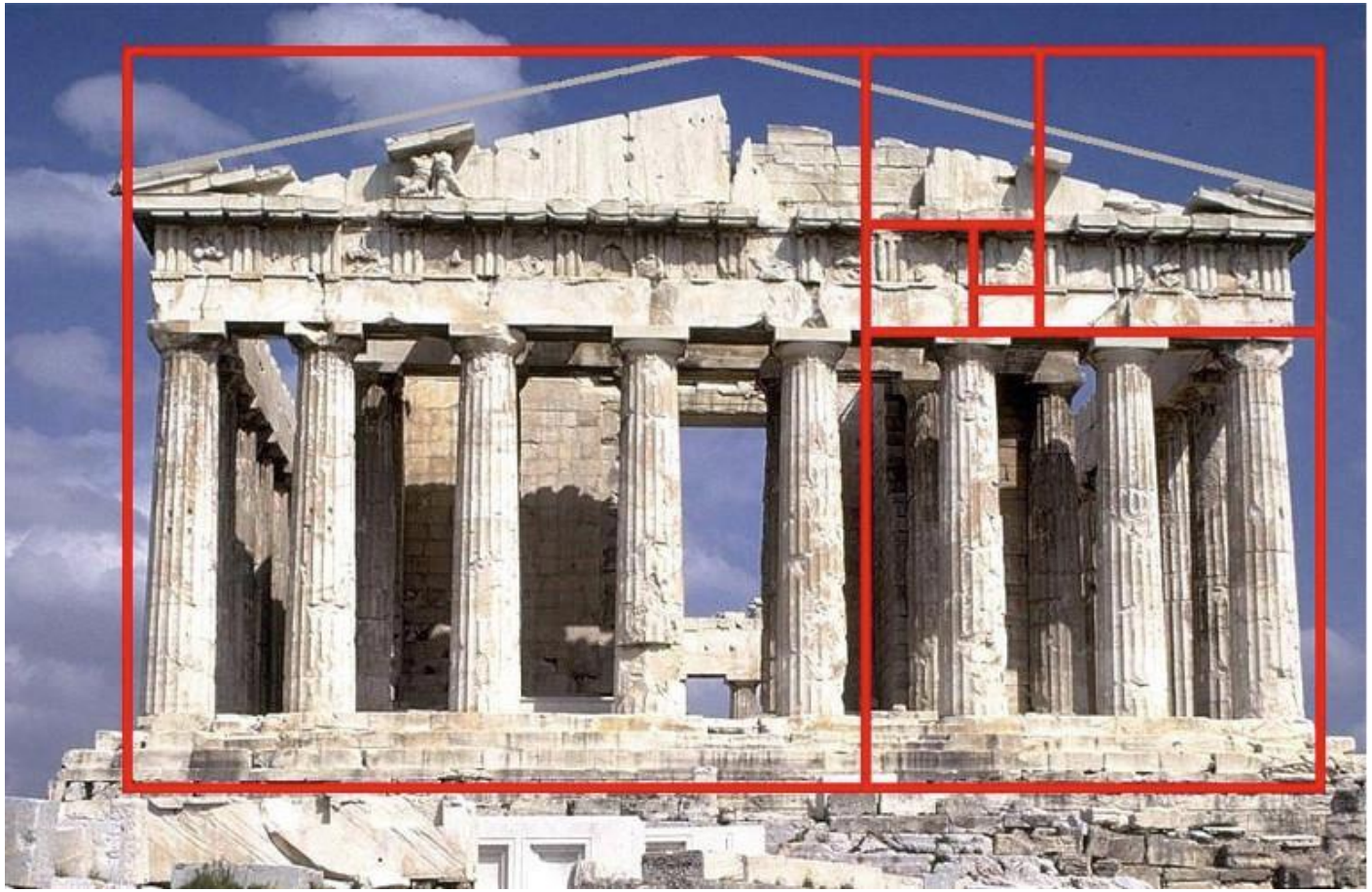
Applications to Fibonnacci – Architecture

Applications to Fibonnacci – Architecture & Arts

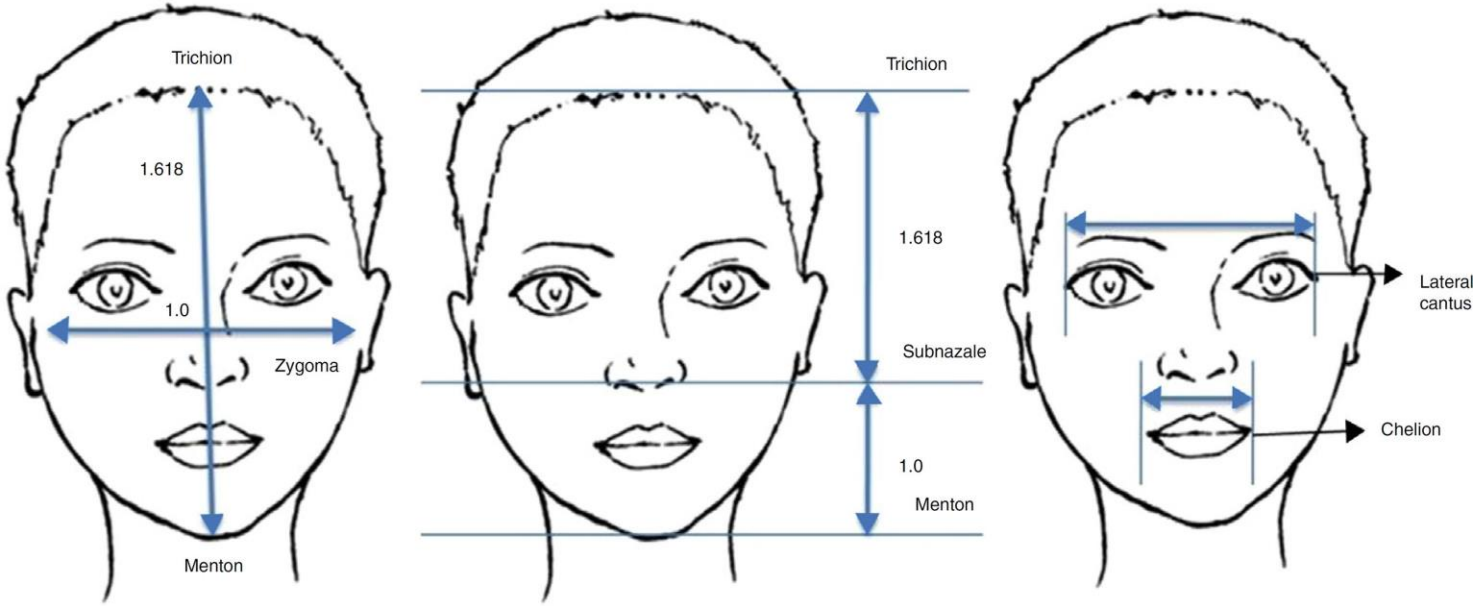


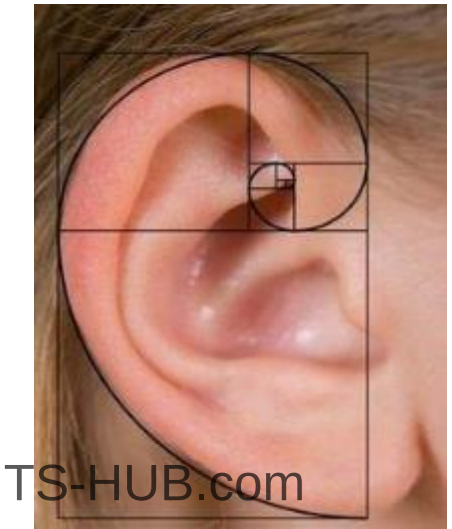


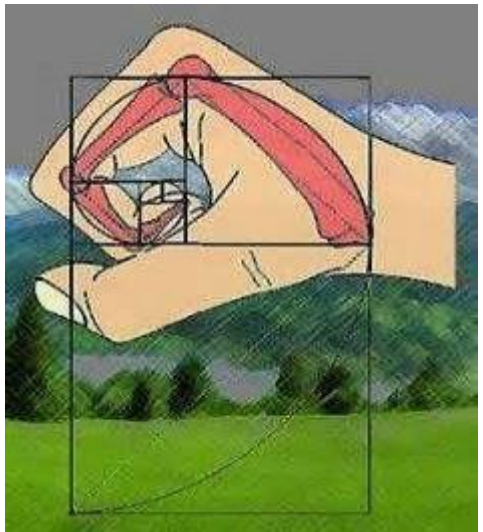


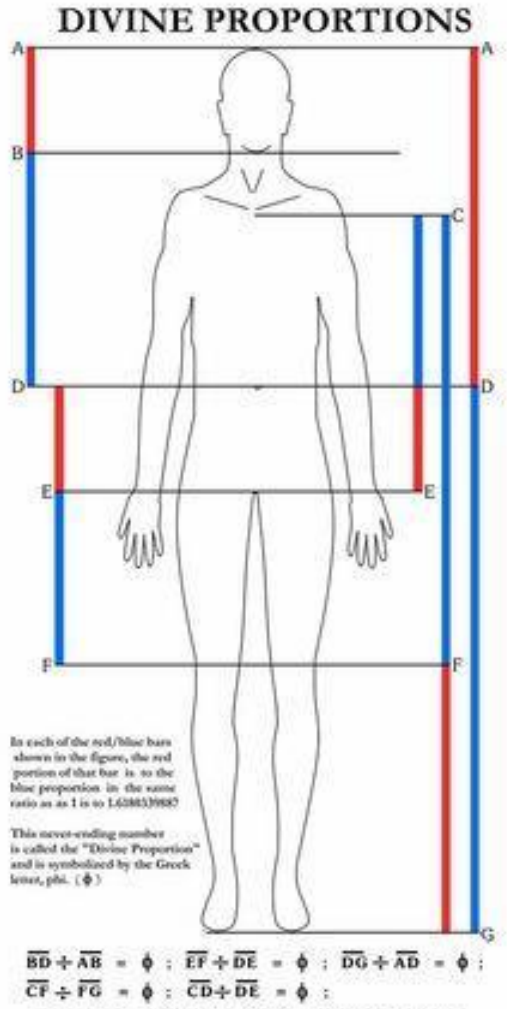


Applications to Fibonnacci – Nature









Fibonnacci series (2)

- Write a recursive implementation of the Fibonnacci.

```
int fibonnacci( int n ){
      int answer;

      if(n == 0 || n == 1)
            answer = n;
      else
            answer = fibonnacci(n - 1) + fibonnacci(n - 2);

      return answer;
}
```

Fibonnacci series (3)