Assignment 1 – Skript-Aufgabe 120 (4 points)

The following function finds an element with a given value \( x \) in a sorted sequence (if there is such an element), using binary search.

```c++
typedef std::vector<int>::const_iterator Cvit;

// PRE: [begin, end) is a valid range, and the elements *p,
// p in [begin, end) are in ascending order
// POST: return value is an iterator p in [begin, end) such
// that *p = x, or the pointer end, if no such iterator
// exists
Cvit bin_search (const Cvit begin, const Cvit end, const int x)
{
    const int n = end - begin; // empty range
    if (n == 0) return end;
    if (n == 1) {
        if (*begin == x)
            return begin;
        else
            return end;
    }
    // n >= 2
    const Cvit middle = begin + n/2;
    if (*middle > x) {
        // x can't be in [middle, end)
        const Cvit p = bin_search (begin, middle, x);
        if (p == middle)
            return end; // x not found
        else
            return p;
    }
    else
        // *middle <= x; we may skip [begin, middle)
    return bin_search (middle, end, x);
}
```

What is the maximum number \( T(n) \) of comparisons between sequence elements and \( x \) that this function performs if the number of sequence elements is \( n \)? Try to find an upper bound on \( T(n) \) that is as good as possible. (You may use the statement of Exercise 121.)
Assignment 2 (4 points)

Rewrite the binary search function from the previous exercise in iterative form. On the course webpage you find the program bin_search_test.cpp, where you can insert and test your code!

Assignment 3 – Skript-Aufgabe 122 (4 points)

Write programs that produce turtle graphics drawings for the following Lindenmayer systems \((\Sigma, P, s)\).

a) \(\Sigma = \{X, Y, +, -\}, s = Y\), and \(P\) given by

\[
X \mapsto Y + X + Y \\
Y \mapsto X - Y - X.
\]

b) Like a), but with the productions

\[
X \mapsto X + Y + +Y - X - -XX - Y + \\
Y \mapsto -X + YY + +Y + X - -X - Y.
\]

For the drawing, use rotation angle \(\alpha = 60\) degrees and interpret both \(X\) and \(Y\) as “move one step forward”.

Assignment 4 – Skript-Aufgabe 123 (4 points)

The Towers of Hanoi puzzle (that can actually be bought from shops) is the following. There are three wooden pegs labeled 1, 2, 3, where the first peg holds a stack of \(n\) disks, stacked in decreasing order of size, see Figure 1.

![Figure 1: The Tower of Hanoi](image)

The goal is to transfer the stack of disks to peg 3, by moving one disk at a time from one peg to another. The rule is that at no time, a larger disk may be on top of a smaller one. For example, we could start by moving the topmost disk to peg 2 (move \((1, 2)\)), then move the next disk from peg 1 to peg 3 (move \((1, 3)\)), then move the smaller disk from peg 2 onto the larger disk on peg 3 (move \((2, 3)\)), etc.

Write a program hanoi.cpp that outputs a sequence of moves that does the required transfer, for given input \(n\). For example, if \(n = 2\), the above initial sequence \((1, 2)(1, 3)(2, 3)\) is already complete and solves the puzzle. Check the correctness of your program by hand at least for \(n = 3\), by manually reproducing the sequence of moves on a piece of paper (or an actual Tower of Hanoi, if you have one).
Challenge – Skript-Aufgabe 126 (Lindenmayer Systems)

Don’t forget to write some recommended parameter setting (for instance the number of iterations) as a comment in your code and also include your name. We will collect your submissions and show a collection of the most beautiful pictures in the lecture.