University of Zurich

Department of Informatics
Database Technology Group
Prof. Dr. Michael Böhlen

Informatics II
Exercise 2

Spring Semester 2022
Week 3
Recursion

Task 1. Short Questions

Solve all of the following subtasks without executing any given code.

(a) Consider the C function below. How many recursive calls will be executed for `rec_fun1(3)`?

```c
int rec_fun1(int n) {
    if (n == 13) {
        return 12;
    }
    else {
        return 11 * rec_fun1(n + 2);
    }
}
```

Answer:

(b) Consider the following function in C
What will be the return value of the call `rec_fun2(3, 0)`?

```c
int rec_fun2(int x, int y) {
    if (x <= 0) {
        y = y + 5;
        return y;
    }
    else {
        int t1 = rec_fun2(x - 1, y + 2);
        int t2 = rec_fun2(x - 2, y + 3);
        return t1 + t2;
    }
}
```

Answer:

(c) Consider the following two C functions:
What will be the output on the console for the call `rec_fun3a(5)`?

```c
void rec_fun3a(int n) {
    if (n == 0) {
        return;
    }
    printf("%d", n);
    rec_fun3b(n - 2);
    printf("%d", n);
}

void rec_fun3b(int n) {
    if (n == 0) {
        return;
    }
    printf("%d", n);
    rec_fun3a(n + 1);
    printf("%d", n);
}
```

Answer:

(d) Consider the following function in C:
Formally describe the set of input values `x` and `y` for which an infinite recursion will occur (i.e. for which the base case is never reached).

```c
int rec_fun4(int x, int y) {
    if (x > y) {
        return x * y;
    }
    else {
        return rec_fun4(x - 1, y);
    }
}
```

Answer:

(e) Is the following statement true or false?
«A recursive function always has to have exactly one base case.»

Answer: ☐ True ☐ False

Task 2. Second Smallest Element
Write a program in C which *recursively* finds the value of the second smallest element in an arbitrary array $A[0..n-1]$ of $n > 1$ mutually distinct, strictly positive integers.
A code skeleton with the code for reading in an array from the user is provided as `task2_skeleton.c`.

Task 3. Blinking Light
Consider a LED which is emitting different blinking patterns. In each second, the LED can either exhibit exactly two short blinks (which will be denoted as **--** in the following examples) or exactly one long blink (denoted as **—**). Hence a blinking pattern will consist of a certain number n of blinks (each either short or long). For example, a blinking pattern consisting of $n = 3$ blinks (regardless whether short or long) can have one of the following 3 configurations:
(-- —), (— --), (— — —)
A blinking pattern consisting of $n = 4$ blinks can be constructed in 5 different ways:
(- - - -), (-- — —), (— -- —), (— — --), (— — — —)
Write a program in C which calculates the number of different blinking patterns which consist of exactly n blinks.
A code skeleton with the code for reading in an integer value is provided as `task3_skeleton.c`.

Task 4. Fractal Circles
Devise a pseudo code algorithm which will produce in a Cartesian coordinate system a picture according to the following rules:
Initially, a circle with radius r_0 is drawn with its centre at the position $(x_0, y_0) = (0, 0)$. At each point of intersection of a circle with the x-axis, another circle is drawn which has half the radius of the circle intersecting the x-axis. No circle with a radius smaller than $r_{min} = 10$ should be drawn. See Figure 1 for an example produced for $r_0 = 256$.
Assume that a subroutine $draw\_circle(pos\_x, pos\_y, radius)$ does already exist and will draw a circle around a centre position at coordinate (pos_x, pos_y) with the radius given as an argument.

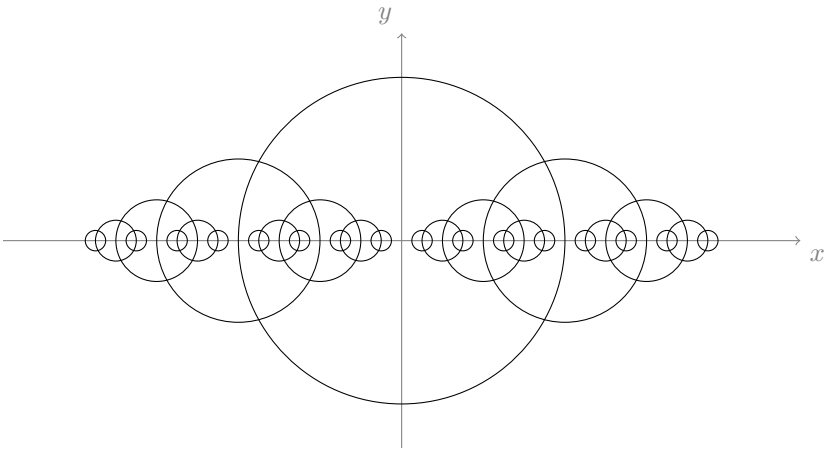

Figure 1: Figure produced according to the given rules for $r_0 = 256$.