

Informatics II

Exercise 2

Spring Semester 2022
Week 3
Recursion

C Tutor! 

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)`?

```

1 int rec_fun1(int n) {
2     if (n == 13) {
3         return 12;
4     }
5     else {
6         return 11 * rec_fun1(n + 2);
7     }
8 }

```

← Base Case (Termination Condition)

recursion call

stored in memory on the stack

1.) $n=3$ 2.) $n=5$ 3.) $n=7$ 4.) $n=9$ 5.) $n=11$ 6.) $n=13$

Answer:

6 total, but only 5 recursion

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

```

1 int rec_fun2(int x, int y) {
2     if (x <= 0) {
3         y = y + 5;
4         return y;
5     }
6     else {
7         int t1 = rec_fun2(x - 1, y + 2);
8         int t2 = rec_fun2(x - 2, y + 3);
9         return t1 + t2;
10    }
11 }

```

1.) $x=3, y=0$ 2) 3)

$y=11$

$x=2, y=2$ $x=1, y=4$ $x=0, y=6$

$x=1, y=3$ $x=-1, y=6$

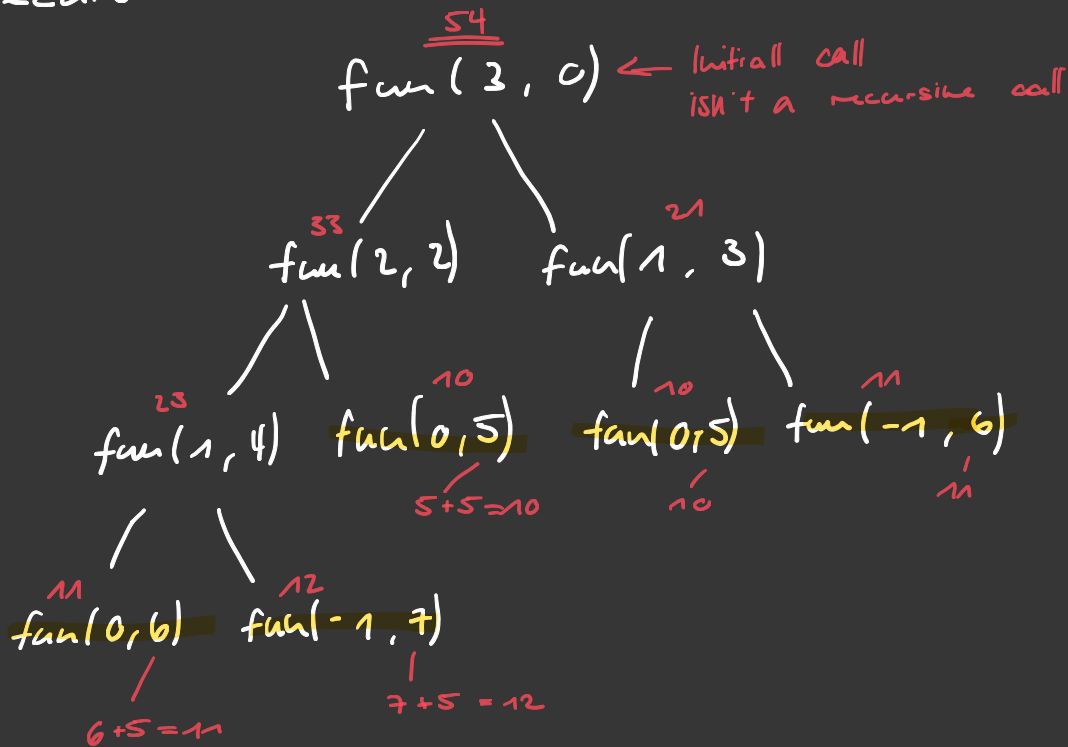
Answer:

22 54

a)

$$\begin{array}{l}
 \text{fun}(3) = 1 \cdot 932 \cdot 612 \\
 | \\
 \text{fun}(5) \quad 1 \cdot 11 \cdot \quad = 1 \cdot 932 \cdot 612 \\
 | \\
 \text{fun}(7) \quad 2 \cdot 11 \cdot 15 \cdot 972 = 175 \cdot 612 \\
 | \\
 \text{fun}(9) \quad 3 \cdot 11 \cdot 1452 = 15 \cdot 972 \\
 | \\
 \text{fun}(11) \quad 4 \cdot 11 \cdot 132 = 1452 \\
 | \\
 \text{fun}(13) \quad 5 \cdot 11 \cdot 12 = 132
 \end{array}$$

b) recursive tree



- (c) Consider the following two C functions:

What will be the output on the console for the call `rec_fun3a(5)`?

```

1 void rec_fun3a(int n) {
2     if (n == 0) {
3         return;
4     }
5     printf("%d", n);
6     rec_fun3b(n - 2);
7     printf("%d", n);
8 }
9
10 void rec_fun3b(int n) {
11     if (n == 0) {
12         return;
13     }
14     printf("%d", n);
15     rec_fun3a(n + 1);
16     printf("%d", n);
17 }

```

Answer:

error?

- (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).

```

1 int rec_fun4(int x, int y) {
2     if (x > y) {
3         return x * y;
4     }
5     else {
6         return rec_fun4(x - 1, y);
7     }
8 }

```

Answer:

 $\forall x < y$

- (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`.

```

int rec_fun(arr[x] = n, len)
    temp 1 = 0
    temp 2 = 0
    if n < temp 1 && n > temp 2 do
        return rec_fun[x+1]
    return

```

c) fun 3a(5) \rightarrow fun 3b(3)

1. print 5
p5

print 3
p3

fun 3b(1) \rightarrow fun 3a(3) \leftarrow fun 3b(4) \rightarrow fun 3a(4)

p1 \rightarrow p3 \rightarrow p2 \rightarrow p4

fun 3a(2) \rightarrow fun 3b(0)

p2 return;

p2

(2)

Base Case: until index == 0

if upper_limit = 0 then

if arr[0] < smallest then
return smallest

else if arr[0] < second_smallest && arr[0] != smallest then

return arr[0]

else

return second_smallest

Pseudo code solution:

```

Algorithm: num_patterns(num_blinks)
if num_blinks == 1 then
  return 1
if num_blinks == 2 then
  return 2
return num_patterns(num_blinks - 1) + num_patterns(num_blinks - 2)

```

add → need at least 1 long blink

Base Cases
 $n = 1 \rightarrow 1$ long blink $\rightarrow 1$ pos.
 $n = 2 \rightarrow 2$ short blinks || 2 long blink $\rightarrow 2$ pos.
 blink(4)
 blink(3) blink(2)
 blink(2) blink(1)
 return 6

University of Zurich

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

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.

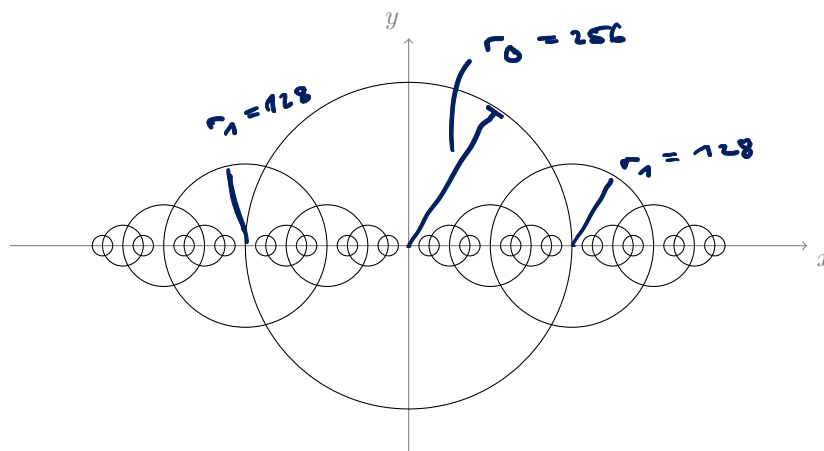


Figure 1: Figure produced according to the given rules for $r_0 = 256$.

```

fractal_circle(pos_x, pos_y, radius):
  if radius > 10 then
    draw_circle(pos_x, pos_y, radius) // only needed once bc. recursive fun. gets called twice with the
    different values
    fractal_circle(pos_x + radius, pos_y, radius/2)
    fractal_circle(pos_x - radius, pos_y, radius/2)

```

Pseudo code solution:

```

Algorithm: fractal_circle(xpos, ypos, radius)

```

```

if radius ≥ 10 then
  draw_circle(xpos, ypos, radius)
  fractal_circle(xpos + radius, ypos, radius/2)
  fractal_circle(xpos - radius, ypos, radius/2)

```