Exercise 4.1 — Fibonacci Instrumentation

Modify the program fib1.c shown in the lecture so that your modified program produces the following output:

```
fib(5) start
  fib(4) start
    fib(3) start
      fib(2) start
        fib(1) start
          fib(1) = 1
        fib(1) = 1
        fib(2) = 1
      fib(1) start
        fib(1) = 1
      fib(3) = 2
    fib(2) start
      fib(1) start
        fib(1) = 1
      fib(0) start
        fib(0) = 0
    fib(2) = 1
  fib(4) = 3
fib(3) start
  fib(2) start
    fib(1) start
      fib(1) = 1
    fib(0) start
      fib(0) = 0
    fib(2) = 1
  fib(1) start
    fib(1) = 1
fib(3) = 2
fib(5) = 5
```

Exercise 4.2 — Simulation of C Program Execution (30% of Midterm 3, 2003)

Simulate execution of the following correct ANSI C program:
- Show all calls to the function f and their arguments and local variables
- Document intermediate states of the array q and indicate where changes are produced
- Show which output is produced, and when

```c
#include <stdio.h>
#define SIZE 2
char q[SIZE+2] = "ae";

void f(int m) { // forward declaration
int main() {
    f(0);
    return 0;
}
void f(int m) {
    char h;
    printf("f(%d) <-- %s\n", m, q);
    if (m >= SIZE) return;
    h = q[m];
    q[m] = q[m+1];
    f(m+1);
    q[m+1] = h+1;
    printf("f(%d) --> %s\n", m, q);
}
```
Exercise 4.3 — Histograms (75% of Midterm 1, 2005)

Assume a sensor that produces int-valued readings in the range from 0 to $MAX\_READING$.

Throughout this question, we will deal with arrays

```c
long int readings[\(MAX\_READING + 1\)]
```

that contain information about the sensor readings in a certain time interval in the following way:

> For \(k \in \{0, \ldots, MAX\_READING\}\), the array element \(readings[k]\) contains the **number of times** the sensor reading produced value \(k\).

**Note:** The solutions of the items are independent of each other.

(a) Assume that the function

```c
int getSensorReading();
```

(which you do not have to implement) obtains an individual reading from the sensor in question.

**Design and implement** the function

```c
void collect(long int readings[], long int number_of_samples);
```

which collects \(number\_of\_samples\) sensor readings into the array \(readings\) such that after the call, \(readings[k]\) contains the **number of times** the sensor reading produced value \(k\) during this call to \(collect\).

Implement \(collect\) in such a way that it waits 0.2 milliseconds between readings; for these delays, use the following library function:

```c
#include <unistd.h>
void usleep(unsigned long usec);
```

The \(usleep\) function suspends execution of the calling process for (at least) \(usec\) microseconds.

(b) Assume that the sensor vendor provided the function \(getSensorReading()\) as a library function without providing source code for it.

What do you have to do to make programs that use \(getSensorReading()\) compile and execute properly? Explain!

(c) **Design and implement** the function

```c
double mean(long int readings[])
```

to calculate with minimal loss of precision the mean of all sensor readings collected in the array \(readings\).

(d) **Design and implement** the function

```c
void display(long int readings[], long int step, int height)
```

to print a histogram representing the contents of \(readings\) to the screen. The histogram is truncated (or padded) to height \(height\).

In this histogram, each element of \(readings\) is turned into one column; each ‘*’ character represents \(step\) sensor readings, and on the top of a column, a ‘^’ character represents less than \(step\) sensor readings (but at least one).

The example histogram to the left should be produced e.g. by calling

```
display(readings, 10, 10)
```

with \(MAX\_READING = 7\) and \(readings\) containing the values 55, 60, 69, 23, 17, 45, 0, 5.