Recursion Exercise 1
Rewrite the following recursive function in iterative form.

```c
unsigned int f (const unsigned int n)
{
    if (n <= 2) return 1;
    return f(n-1) + 2 * f(n-3);
}
```
Recursion Exercise 1

Solution below uses just 4 variables. Other solutions are of course also possible.

```c
unsigned int f_it (const unsigned int n)
{
    if (n <= 2) return 1;
    unsigned int a = 1;  // f(0)
    unsigned int b = 1;  // f(1)
    unsigned int c = 1;  // f(2)
    for (unsigned int i = 3; i < n; ++i) {
        const unsigned int a_prev = a;  // f(i-3)
        a = b;                           // f(i-2)
        b = c;                           // f(i-1)
        c = b + 2 * a_prev;             // f(i)
    }
    return c + 2 * a;                 // f(n-1) + 2 * f(n-3)
}
```

(From: Script Exercise 128)
Recursion Exercise 2
Rewrite the following recursive function in iterative form.

```c
unsigned int f (const unsigned int n)
{
    if (n == 0) return 1;
    return f(n-1) + 2 * f(n/2);
}
```
Recursion Exercise 2

Solution below stores intermediate results in a vector. Other solutions are of course also possible.

```cpp
unsigned int f_it (const unsigned int n)
{
    if (n == 0) return 1;
    std::vector<unsigned int> f_values(n + 1, 0);
    f_values[0] = 1;
    for (unsigned int i=1; i<=n; ++i)
        f_values[i] = f_values[i-1] + 2 * f_values[i/2];
    return f_values[n];
}
```

(From: Script Exercise 129)