```
// Fibonacci in F#

// From “Expert F#”
// by Don Syme, Adam Granicz, Antonio Cisternino
// free and available at http://www.expert-fsharp.com

open System.ComponentModel
open System.Windows.Forms

let worker = new BackgroundWorker()
let numIterations = 1000

worker.DoWork.Add(fun args ->
    let rec computeFibonacci resPrevPrev resPrev i =

        // Compute the next result
        let res = resPrevPrev + resPrev

        // At the end of the computation and write the result
        if i = numIterations then
            args.Result <- box res
        else
            // Compute the next result
            computeFibonacci resPrev res (i+1)

   computeFibonacci 1 1 2)

worker.RunWorkerCompleted.Add(fun args ->
    MessageBox.Show(sprintf "Result = %A" args.Result) |> ignore)

// Execute the worker
worker.RunWorkerAsync()
```

```
// Fibonacci in cilk    (from the cilk manual, MIT)

#include <cilk-lib.cilkh>
#include <stdlib.h>
#include <stdio.h>

int fib_ser(int n)
{
     if (n < 2)
          return (n);
     else {
          int x, y;
          x = fib_ser(n - 1);
          y = fib_ser(n - 2);
          return (x + y);
     }
}

cilk int fib(int n)
{
     if (n < 2)
          return (n);
     else if (n < 30)
          return fib_ser(n);
     else {
          int x, y;
          x = spawn fib(n - 1);
          y = spawn fib(n - 2);
          sync;
          return (x + y);
     }
}

cilk int main(int argc, char *argv[])
{
     int n, result;

     if (argc != 2) {
          fprintf(stderr, "Usage: fib [<cilk options>] <n>\n");
          Cilk_exit(1);
     }
     n = atoi(argv[1]);
     result = spawn fib(n);
     sync;

     printf("Result: %d\n", result);
     return 0;
}
```