実験(S-8) 必須課題 1-3

情報学群 情報科学類

202113564 三村潤之介

Fibonacci 数を計算する関数 fib を定義せよ。ただし、
fib(1) = 1
fib(2) = 1
fib(n+2) = fib(n) + fib(n+1)　if n>0
である。

Fibonacci 数を計算する際、定義に基づいて再帰的に求める方法を最初に思いついたが、オプショナル課題の取り組みとして、効率を改善したプログラムを考えることにした。前述したアルゴリズムの効率の悪さは、Fibonacci 数を重複して求めていることに起因する。そのため、直前 2 つの Fibonacci 数を引数として保持しながら目的の数を求めることで重複して求めないようにした。

正の整数 n が与えられると、n 番目の素数を計算する関数 prime を定義せよ。たと えば、
prime(1) = 2
prime(3) = 5
prime(5) = 11
である。

n 番目の素数を求めるには、i を 2 からカウントアップしていって、i が素数である状態が n 回起きたときの i を出力すれば良いと考えた。i が素数であることは prime_check (int -> int -> bool)として関数を作って判定することにした。こういったカウント変数を Fibonacci 数の実装と同様に、引数として保持して再帰的に呼び出すことをしているが、およそ非効率ではないだろうかと考えてしまう。実験を通して、再帰関数の実装に慣れていきたい。

文字列 s1 と文字列 s2 が与えられたとき、s1 に s2 が含まれているな らその位置を(複数含まれているなら、その一番左側の位置を)返し、 含まれていないなら -1 を返す関数 substring を定義せよ。 たとえば、
substring "abcabc" "abc" = 0
substring "abcabc" "bca" = 1
substring "abcabc" "bcd" = -1
である。

s1 と s2 の文字列の文字をそれぞれ参照していって比較する必要があるため、カウント変数が 2 つ必要だと考えた。s2 の文字数回分、比較が true であれば、s1 のカウント変数をもとに位置を計算(counter_s1 – s2.length)して出力する。前までの課題と同じく、2 つのカウント変数を引数として持つように変換して、サブ関数の中でメインとなる処理を行う。実装において、やはり不慣れであることから書きたい処理に制限が出てしまっていると感じる。特に C や Java の逐次処理などの概念は Ocaml では存在しないのか、形が大きく変わっているのかして、実装で用いる事ができない。実験を通して解決できればと思う。