COL100

By Vansh Gupta (2019CH10143)

Assignment 1

Q1. Computing factorial of a given integer using both recursive and iterative procedures

A1.

Recursive:

Assignment 2

(1) Factorial - Recursive

$f : \mathbb{N} \to \mathbb{N} \qquad f(x) = x!$

Proof:

Induction Basis:

$f(0) = 1$

Induction Hypothesis:

$f(n-1) = (n-1)! \ \forall n \in \mathbb{N}$

Induction Step:

$f(n) = n * f(n-1)$ ~ by Algo

$= n(n-1)!$ ~ by induc^n hypothesis

$= n!$ ⇒ Function computes $n!$ for $n \in \mathbb{N}$

Algo:

```
fun f(n) =
 if n=0 then 1
 else
 n*f(n-1);
```

Iterative:

Factorial Iterative

Proof:

TST: $fact\_iter(m, f, c) = f * \prod_{i=c+1}^{m} i$

Invariant: $(c_0 \le m \le c) \wedge (f = f_0 * \prod_{i=c_0+1}^{c} i) \wedge$

$(f_0 * \prod_{i=c_0+1}^{m} i = f * \prod_{i=c+1}^{m} i)$

~~Basis~~ Proof by PMI

$i, c_0, m, c \in \mathbb{N}$

Basis

Initially for $m - c = 0$

$fact\_iter(m, f, c) = f * \prod_{i=m+1}^{m} i = f * 1$

($\because$ since $\prod$ contains empty set $\therefore$ initial value = final value, $\therefore 1f$ is returned)

Hypothesis:

$\exists\, k \in \mathbb{N}$ s.t $k = m - c$

$fact\_iter(m, f, c) = f * \prod_{i=c+1}^{m} i$

Induction step:

for $k = m - c - 1$

i.e for $m - c = k + 1$

$fact\_iter(m, f, c) = fact\_iter(m, f * (c+1), c+1)$

$= (f * (c+1)) * \prod_{i=c+2}^{m} i$

$= f * ((c+1) * \prod_{i=c+2}^{m} i) = f * \prod_{i=c+1}^{m} i$

$\therefore$ By Substitution

$$\text{fact-iter}(m,1,0) = 1 \times \prod_{i=1}^{m} i = 1 \times 1 \times 2 \times 3 \times \_\_\_ \times m$$

$$= m!$$

Hence proved

Q2. Computing x^n. Write both recursive and iterative versions.

A2.
Recursive:

(2) Power Recursive

$f : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ $\quad f(x, m) = x^m$

Proof:

Induction Basis:

$f(0) = 1$

I.H: $f(x, m-1) = x^{m-1}$

for some $x, m-1 \in \mathbb{N}$

Induction Step:

$f(x, m) = x * f(x, m-1)$ - by Algo

$= x \times x^{m-1}$ ~ by Def.

$= x^m$

$\Rightarrow$ func^n computes x^m for $x, m \in \mathbb{N}$

Algo:

fun(x, m) =

if m=0 then 1

else

~~x*f(x,m-1)~~ x * f(x, m-1);

Iterative:

Power Function _ Iterative :

```
fun f(x, n) =
    let fun p_iter (p, c, n) =
            if ~~p=0~~ n=0 then p
            else p_iter (p*c, c, n-1)
    in p_iter(1, x, n)
end;
```

Proof :

TST : $p\_iter\ (p, c, n) = p^* c^n$ s.t when we call $p\_iter(1, x, n)$

it returns the value x^n.

~~Basis~~ Invariant : $(n_0 \geq n \geq 0) \wedge (p^* c^n = p_0^* c^{n_0})$

Basis : Initially for $n=0$

By algo, $p\_iter\ (p, c, 0) = p = p \times c^0$

$\Rightarrow$ The basis holds

Induction Hypotheses :

for $n = i \geq 0$

$p\_iter\ (p, c, i) = p^* c^i$

Induction step :

for $n = i+1$

$i+1-1 = i$

$p\_iter\ (p, c, n) = p\_iter\ (p^* c, c, i)$

$= \cancel{(p * c)} \quad (p \times c) \times (c^i) \quad$ [By Algo]

$= p \times (c^i \times c) = p \times c^{i+1}$

Hence proved p_iter $(p, c, m) = p * c^m$

Now,

When p-iter is called is with arguments 1, x and n

the funcn computes the value x^n which is desired

Q3. Computing the n^{th} fibonacci number. First use the algorithm given by the following functional description: fib(1) = 1; fib(2) = 1; fib(n) = fib(n -1)+fib(n-2) for n > 2. Also develop iterative algorithms for the same problem.

A3.
Since Fibonacci Sequence is defined inductively, therefore its proof is trivial.

Q4. The integer square root of n is the integer k such that k^2 <= n < (k+1)^2. The integer square root can be computed using the following inductive process:

- •Compute the integer square root I of m = n div 4 recursively. We then have that i^2 <= m < (i+1)^2.
- •Since m and I are integers we have that (m+1) <= (i+1)^2. We thus have (2i)^2 <= 4m <= n < 4m + 4 <= (2i + 2)^2. Hence we have that the integer square root of n is either 2i or 2i+1.
- •Write a recursive ML program corresponding to the above algorithm. Indicate the type of the function and derive the number of steps required.

A4. The type of function is recursive and the total steps it takes is ($\log_4(n) + 2$)

$$
\begin{aligned}
T(4^n) &= T(4^{(n-1)}) + 1 \\
&= T(4^{(n-2)}) + 2 \\
&\;\;. \\
&\;\;. \\
&\;\;. \\
&\;\;. \\
&= T(1) + n \\
&= T(0) + n + 1
\end{aligned}
$$

Hence number of steps = $\log_4(n) + 2$

Proof:

(1) Integer Sq. root $f: \mathbb{N} \to \mathbb{N}$ (Recursive)

$f(x) = m$ s.t $m^2 \le x < m+1$

~~Proof~~ Algo:

fun $f(m)$:

if $m = 0$ then 0

else if $(2 * f(m \text{ div } 4) + 1) * (2 * f(m \text{ div } 4) + 1) <= m$ then

$2 * f(m \text{ div } 4) + 1$

else $2 * f(m \text{ div } 4)$

Proof:

~~Basis~~ (Let k be square root of m div 4

i.e $k^2 \le (m \text{ div } 4) < (k+1)^2$) $k, m \in \mathbb{N}$

Induction Hypothesis:

Basis: $f(0) = 0$ i.e $sqrt(0) = 0$

Proof. $0^2 \le 0 < (0+1)^2$ $0 \le 0 < 1$

Induction step

Case I: $m < (2k+1)^2$

From IH, $k^2 \le (m \text{ div } 4)$

$\Rightarrow 4k^2 \le (m \text{ div } 4) * 4$

~~for some $m \in \mathbb{N}$ $(m \text{ div } 4) * 4$ can give values $m, m+1, m+2, m+3$~~

$\therefore (m \text{ div } 4) * 4 \le m$

m can be $4m, 4m+1, 4m+2, 4m+3$

$m \text{ div } 4 = m$ $(m \text{ div } 4) * 4 = 4m$ $m < (2k+1)^2$

$\Rightarrow (m \text{ div } 4) * 4 \le m$

$\therefore 4k^2 \le m$ $(2k)^2 \le m$

& since natural numbers are considered by the specific case

$(2k)^2 \le m < (2k+1)^2$

Case II: $m \ge (2k+1)^2$

From IH, $(m \div 4) < (k+1)^2$ where $\div$ is div function

Since we are considering natural numbers

$1 + (m \div 4) \le (k+1)^2$

$4 + 4 * (n \div 4) \leq 4(k+1)^2$

$4 + 4 * (n \div 4) \leq (2k+2)^2$

$\exists\ m \in \mathbb{N}$ s.t $n \in \{4m, 4m+1, 4m+2, 4m+3\}$

$4 * (n \div 4) = 4m$

$\Rightarrow LHS = 4m + 4$

i.e $4 + 4 * (n \div 4) > n$

$\Rightarrow n < 4 + 4 * (n \div 4) \leq (2k+2)^2$

$n \leq (2k+2)^2$

By initial condⁿ, $(2k+1)^2 \leq n < (2k+2)^2$

$\therefore$ $2k+1$ is integer sqr root

Q5. Study the problem of computing *perfect numbers* from the Lecture notes Example 3.13 and implement the ML program. Also study the following discussion on *scope rules*. You will be questioned on this problem during the demonstration.

A5. <Studied in class and Implemented the program and submitted>