Assignment 2

Nathan Ellis, Rishad Mahbub

May 2, 2018

1 Task 1

Define a function c: $\mathbb{R} \longrightarrow \mathbb{R}$ as

$$c(n) = \lfloor \log_{10} n \rfloor$$

Define the set of all primes $\mathbb{P}$ as

$$\mathbb{P} = \{ \ x \in \mathbb{N} \mid \forall (1 < i < x) \ . \ xmod(i) \neq 0 \ \}$$

Define the set of all emirps $\mathbb{E}$ as

$$\mathbb{E} = \{ \ x \in \mathbb{N} \mid x = \sum_{i=0}^{c(x)} S_i 10^{c(x)-i} \land r = \sum_{i=0}^{c(x)} S_i 10^i \ \land x \in \mathbb{P} \land r \in \mathbb{P} \land x \neq r \ \}$$

where S is some constant.

Define a function f: $\mathbb{N} \longrightarrow \mathbb{N}$ as

$$f(x) = \begin{cases} 0 & \textbf{if } y \notin \mathbb{E} \\ 1 & \textbf{if } y \in \mathbb{E} \end{cases}$$

We specify our function as required by the assignment 2 specification as

$$n : \left[\text{ emirpNumber} > 0, \ f(n) = 1 \land \text{emirpNumber} = \textstyle\sum_{k=0}^{n} f(k) \ \right]$$

As the precondition states, the user input number called *emirpNumber* is greater than 0. The program specification then states that n is changed such that the number of emirps between 0 and n inclusive is equal to *emirpNumber* and n is itself an emirp.

We also will specify a helper function as

$$x : \left[\ y > 0, \ x = f(y) \ \right]$$

Which takes input y and sets x according to the function f.

2 Task 2

We will use two functions and so derive both using refinement calculus.

2.1 Emirp Derivation

We start with a spec of the procedure EMIRP.

$$\textbf{proc}\ \text{EMIRP}(\textbf{value}\ n : \mathbb{N},\ \textbf{value}\ \text{emirpNumber} : \mathbb{N}) \cdot$$

$$n : \left[\ \text{emirpNumber} > 0,\ \textstyle\sum_{k=1}^{n} f(k) = \text{emirpNumber} \wedge f(n) = 1\ \right]$$

$\sqsubseteq$ $\langle$**i-loc**$\rangle$

$$\llcorner n, m, \text{emirpsCounted} : \left[\ \text{emirpNumber} > 0,\ \textstyle\sum_{k=1}^{n} f(k) = \text{emirpNumber} \wedge f(n) = 1\ \right]\lrcorner(1)$$

(1) $\sqsubseteq$ $\langle$**seq - establishing a loop where I=**$\sum_{k=1}^{n} f(k) = \text{emirpsCounted} \wedge f(n) = m \wedge n > 0$$\rangle$

$$\llcorner n, m, \text{emirpsCounted} : \left[\begin{array}{l} \text{emirpNumber} > 0, \\ \sum_{k=1}^{n} f(k) = \text{emirpsCounted} \wedge f(n) = m \wedge n > 0 \end{array}\right]; \lrcorner(2)$$

$$\llcorner n, m, \text{emirpsCounted} : \left[\begin{array}{l} \sum_{k=1}^{n} f(k) = \text{emirpsCounted} \wedge f(n) = m \wedge n > 0, \\ \sum_{k=1}^{n} f(k) = \text{emirpsCounted} \wedge f(n) = m \wedge n > 0 \\ \wedge\ \text{emirpsCounted} = \text{emirpNumber} \wedge f(n) = 1 \end{array}\right]; \lrcorner(3)$$

$$\llcorner n, m, \text{emirpsCounted} : \left[\begin{array}{l} \sum_{k=1}^{n} f(k) = \text{emirpsCounted} \wedge f(n) = m \wedge n > 0 \\ \wedge\ \text{emirpsCounted} = \text{emirpNumber} \wedge f(n) = 1, \\ \sum_{k=1}^{n} f(k) = \text{emirpNumber} \wedge f(n) = 1 \end{array}\right] \lrcorner(4)$$

(2) $\sqsubseteq$ $\langle$**ass - justification below in 2.1.1**$\rangle$

$n := 1;\ \text{emirpsCounted} := 0;\ m := 0$

(3) $\sqsubseteq$ $\langle$**while**$\rangle$

while $(\text{emirpsCounted} \neq \text{emirpNumber} \vee f(n) \neq 1)$ **do**

$$\llcorner n, m, \text{emirpsCounted} : \left[\begin{array}{l} \sum_{k=1}^{n} f(k) = \text{emirpsCounted} \wedge f(n) = m \wedge n > 0 \\ \wedge(\text{emirpsCounted} \neq \text{emirpNumber} \vee f(n) \neq 1), \\ \sum_{k=1}^{n} f(k) = \text{emirpsCounted} \wedge f(n) = m \wedge n > 0 \end{array}\right] \lrcorner(5)$$

od

(4) $\sqsubseteq$ $\langle$**skip - justification below in 2.1.2**$\rangle$

skip;

(5) $\sqsubseteq$ $\langle$**seq2**$\rangle$

$$\llcorner n : \left[\begin{array}{l} \sum_{k=1}^{n} f(k) = \text{emirpsCounted} \wedge f(n) = m \wedge n > 0 \\ \wedge(\text{emirpsCounted} \neq \text{emirpNumber} \vee f(n) \neq 1), \\ \sum_{k=1}^{n-1} f(k) = \text{emirpsCounted} \wedge f(n-1) = m \wedge n > 0 \end{array}\right]; \lrcorner(6)$$

$$\llcorner n, m, \text{emirpsCounted} : \left[\begin{array}{l} \sum_{k=1}^{n-1} f(k) = \text{emirpsCounted} \wedge f(n-1) = m \wedge n > 0, \\ \sum_{k=1}^{n} f(k) = \text{emirpsCounted} \wedge f(n) = m \wedge n > 0 \end{array}\right] \lrcorner(7)$$

(6) $\sqsubseteq$ $\langle$**ass - justification below in 2.1.3**$\rangle$

$n := n + 1;$

(7) $\sqsubseteq$ $\langle$**seq2**$\rangle$

$\llcorner m : \left[\begin{array}{l} \sum_{k=1}^{n-1} f(k) = \text{emirpsCounted} \wedge f(n-1) = m \wedge n > 0, \\ \sum_{k=1}^{n-1} f(k) = \text{emirpsCounted} \wedge f(n) = m \wedge n > 0 \end{array} \right] ;\lrcorner_{(8)}$

$\llcorner n, m, \text{emirpsCounted} : \left[\begin{array}{l} \sum_{k=1}^{n-1} f(k) = \text{emirpsCounted} \wedge f(n) = m \wedge n > 0, \\ \sum_{k=1}^{n} f(k) = \text{emirpsCounted} \wedge f(n) = m \wedge n > 0 \end{array} \right] \lrcorner_{(9)}$

(8) $\sqsubseteq$ $\langle$**s-post - justification below in 2.1.4**$\rangle$

$m : \left[\begin{array}{l} \sum_{k=1}^{n-1} f(k) = \text{emirpsCounted} \wedge f(n-1) = m \wedge n > 0, \\ f(n) = m \end{array} \right];$

$\sqsubseteq$ $\langle$**w-pre - justification below in 2.1.5**$\rangle$

$m : [\ n > 0,\ f(n) = m\ ];$

$\sqsubseteq$ $\langle$**proc - defined and derived below in section 2.2**$\rangle$

$\text{isEmirp}(m, n);$

(9) $\sqsubseteq$ $\langle$**c-frame of n, m**$\rangle$

$\text{emirpsCounted} : \left[\begin{array}{l} \sum_{k=1}^{n-1} f(k) = \text{emirpsCounted} \wedge f(n) = m \wedge n > 0, \\ \sum_{k=1}^{n} f(k) = \text{emirpsCounted} \wedge f(n) = m \wedge n > 0 \end{array} \right]$

$\sqsubseteq$ $\langle$**if**$\rangle$

if m=1 **then**

$\llcorner \text{emirpsCounted} : \left[\begin{array}{l} \sum_{k=1}^{n-1} f(k) = \text{emirpsCounted} \wedge f(n) = m \wedge n > 0 \wedge m = 1, \\ \sum_{k=1}^{n} f(k) = \text{emirpsCounted} \wedge f(n) = m \wedge n > 0 \end{array} \right] \lrcorner_{(10)}$

else

$\llcorner \text{emirpsCounted} : \left[\begin{array}{l} \sum_{k=1}^{n-1} f(k) = \text{emirpsCounted} \wedge f(n) = m \wedge n > 0 \wedge m = 0, \\ \sum_{k=1}^{n} f(k) = \text{emirpsCounted} \wedge f(n) = m \wedge n > 0 \end{array} \right] \lrcorner_{(11)}$

fi

(10) $\sqsubseteq$ $\langle$**ass - justification below in 2.1.6**$\rangle$

$\text{emirpsCounted} := \text{emirpsCounted} + 1;$

(11) $\sqsubseteq$ $\langle$**skip - justification below in 2.1.7**$\rangle$

skip;

There are still some outstanding proofs in the above derivation. We will discharge them below in the following subsubsections.

2.1.1 Proof of $(2) \sqsubseteq n := 1;$ **emirpsCounted** $:= 0; m := 0$

We need to prove the validity of

$$pre(2) \Rightarrow (post(2))[^1/_n][^0/_{\text{emirpsCounted}}][^0/_m]$$

We can prove this from the bottom up.

$$\begin{array}{ll}
& \text{TRUE} \\
\Leftrightarrow & \quad \langle \text{All conjuncts below are logically true} \rangle \\
& 0 = 0 \wedge 0 = 0 \wedge 1 > 0 \\
\Leftrightarrow & \quad \langle \text{Using definition of f(1) to determine f(1)=0 as 1 is clearly not an emirp} \rangle \\
& f(1) = 0 \wedge f(1) = 0 \wedge 1 > 0 \\
\Leftrightarrow & \quad \langle \text{Expanding sigma of first conjunct below} \rangle \\
& \sum_{k=1}^{1} f(k) = 0 \wedge f(1) = 0 \wedge 1 > 0 \\
\Leftrightarrow & \quad \langle \text{Definitions and performing substituitions} \rangle \\
& (post(2))[^1/_n][^0/_{\text{emirpsCounted}}][^0/_m]
\end{array}$$

Hence any true precondition implies the given postcondition.

2.1.2 Proof of $(4) \sqsubseteq skip;$

We need to prove the validity of

$$pre(4) \Rightarrow (post(4))[^n/_{n_0}][^{\text{emirpsCounted}}/_{\text{emirpsCounted}_0}][^m/_{m_0}]$$

We can prove from the top down:

$$\begin{array}{ll}
& pre(4) \\
\Leftrightarrow & \quad \langle \text{Definition} \rangle \\
& \sum_{k=1}^{n} f(k) = \text{emirpsCounted} \wedge f(n) = m \wedge n > 0 \ \wedge \text{emirpsCounted} = \text{emirpNumber} \wedge f(n) = 1 \\
\Rightarrow & \quad \langle \text{Discarding true conjuncts as A} \wedge \text{B} \Rightarrow \text{A} \rangle \\
& \sum_{k=1}^{n} f(k) = \text{emirpsCounted} \wedge \text{emirpsCounted} = \text{emirpNumber} \ \wedge f(n) = 1 \\
\Leftrightarrow & \quad \langle \text{Substituiting emirpsCounted for emirpNumber as emirpsCounted} = \text{emirpNumber} \rangle \\
& \sum_{k=1}^{n} f(k) = \text{emirpNumber} \wedge f(n) = 1
\end{array}$$

2.1.3 Proof of $(6) \sqsubseteq n := n+1;$

We need to prove the validity of

$$pre(6) \Rightarrow (post(6))[^{n+1}/_{n}]$$

We can prove this from the top down initially:

$$\begin{aligned}
& pre(6) \\
\Leftrightarrow \quad & \langle \text{Definition} \rangle \\
& \sum_{k=1}^{n} f(k) = \text{emirpsCounted} \wedge f(n) = m \wedge n > 0 \\
& \wedge (\text{emirpsCounted} \neq \text{emirpNumber} \vee f(n) \neq 1) \\
\Rightarrow \quad & \langle \text{Discarding true conjuncts as A} \wedge \text{B} \Rightarrow \text{A} \rangle \\
& \sum_{k=1}^{n} f(k) = \text{emirpsCounted} \wedge f(n) = m \wedge n > 0 \\
\Rightarrow \quad & \langle \text{Using the fact that n>0} \Rightarrow \text{n>-1} \rangle \\
& \sum_{k=1}^{n} f(k) = \text{emirpsCounted} \wedge f(n) = m \wedge n > -1
\end{aligned}$$

The we finish from the bottom up using equivalences.

$$\begin{aligned}
& \sum_{k=1}^{n} f(k) = \text{emirpsCounted} \wedge f(n) = m \wedge n > -1 \\
\Leftrightarrow \quad & \langle \text{Simplification of the below} \rangle \\
& \sum_{k=1}^{n+1-1} f(k) = \text{emirpsCounted} \wedge f(n+1-1) = m \wedge n+1 > 0 \\
\Leftrightarrow \quad & \langle \text{Defintion of post(6) and substituitions} \rangle \\
& (post(6))[^{n+1}/_{n}]
\end{aligned}$$

2.1.4 Proof of (s-post) $pre(8)[^{m_0}/_{m}] \wedge f(n) = m \Rightarrow (post(8))$

We can prove this from the top down initially:

$$\begin{aligned}
& pre(8)[^{m_0}/_{m}] \wedge f(n) = m \\
\Leftrightarrow \quad & \langle \text{Definition and substituition} \rangle \\
& \sum_{k=1}^{n-1} f(k) = \text{emirpsCounted} \wedge f(n-1) = m_0 \wedge n > 0 \wedge f(n) = m
\end{aligned}$$

$$\begin{aligned}
\Rightarrow \quad & \langle \text{Discard f(n-1)=m}_0 \text{ as A} \wedge \text{B} \Rightarrow \text{A}\rangle \\
& \sum_{k=1}^{n-1} f(k) = \text{emirpsCounted} \wedge f(n) = m \wedge n > 0 \\
\Leftrightarrow \quad & \langle \text{Definition} \rangle \\
& post(8)
\end{aligned}$$

2.1.5 Proof of (w-pre) $\sum_{k=1}^{n-1} f(k) = \textbf{emirpsCounted} \wedge f(n-1) = m \wedge n > 0 \Rightarrow n > 0$

We can prove this simply by discarding true conjuncts:

$$\begin{aligned}
& \sum_{k=1}^{n-1} f(k) = \text{emirpsCounted} \wedge f(n-1) = m \wedge n > 0 \\
\Rightarrow \quad & \langle \text{Discard all but last conjunct as A} \wedge \text{B} \Rightarrow \text{A}\rangle \\
& n > 0
\end{aligned}$$

2.1.6 Proof of $(10) \sqsubseteq emirpsCounted := \textbf{emirpsCounted} + 1;$

We need to prove the validity of

$$pre(10) \Rightarrow (post(10))[^{\text{emirpsCounted+1}}/_{\text{emirpsCounted}}]$$

We can prove this from the top down:

$$\begin{aligned}
& pre(10) \\
\Leftrightarrow \quad & \langle \text{Definition} \rangle \\
& \sum_{k=1}^{n-1} f(k) = \text{emirpsCounted} \wedge f(n) = m \wedge n > 0 \wedge m = 1 \\
\Rightarrow \quad & \langle \text{Using the fact the f(n) = m to substitute m in the last conjunct} \rangle \\
& \sum_{k=1}^{n-1} f(k) = \text{emirpsCounted} \wedge f(n) = m \wedge n > 0 \wedge f(n) = 1 \\
\Leftrightarrow \quad & \langle \text{Absorbing f(n) = 1 into the sigma conjunct} \rangle \\
& \sum_{k=1}^{n} f(k) = \text{emirpsCounted} + 1 \wedge f(n) = m \wedge n > 0 \\
\Leftrightarrow \quad & \langle \text{Definition and substitution} \rangle \\
& (post(10))[^{\text{emirpsCounted+1}}/_{\text{emirpsCounted}}]
\end{aligned}$$

2.1.7 Proof of $(11) \sqsubseteq skip;$

We need to prove the validity of

$$pre(11) \Rightarrow (post(11))[^{\text{emirpsCounted}}/_{\text{emirpsCounted}_0}]$$

Before we start this proof, we realize that this else branch implies m $\neq$ 1.
However, the precondition gives m = f(n). The f() function only maps to values 0 and 1. Hence if m is not equal to 1, it implies m is equal to 0.
Going back to the proof, we can prove this from the top down:

$$pre(4)$$

$\Leftrightarrow$ ⟨Definition⟩

$$\sum_{k=1}^{n-1} f(k) = \text{emirpsCounted} \wedge f(n) = m \wedge n > 0 \wedge m = 0$$

$\Rightarrow$ ⟨Using the fact that f(n) = m to substituite m in the last conjunct⟩

$$\sum_{k=1}^{n-1} f(k) = \text{emirpsCounted} \wedge f(n) = m \wedge n > 0 \wedge f(n) = 0$$

$\Leftrightarrow$ ⟨Absorbing f(n) = 0 into the sigma of the first conjunct. No affect to sum as f(n) = 0⟩

$$\sum_{k=1}^{n} f(k) = \text{emirpsCounted} \wedge f(n) = m \wedge n > 0$$

$\Leftrightarrow$ ⟨Definition and subtituition⟩

$$(post(11))[^{\text{emirpsCounted}}/_{\text{emirpsCounted}_0}]$$

2.2 IsEmirp Derivation

For this program specification of ISEMIRP, we use our definition

$$\mathbb{E} = \{\ x \in \mathbb{N} \mid x = \sum_{i=0}^{c(x)} S_i 10^{c(x)-i} \wedge r = \sum_{i=0}^{c(x)} S_i 10^i\ \ \wedge x \in \mathbb{P} \wedge r \in \mathbb{P} \wedge x \neq r\ \}$$

where S is some constant.

We will also use our definition of function f:

$$f(x) = \begin{cases} 0 & \textbf{if } y \notin \mathbb{E} \\ 1 & \textbf{if } y \in \mathbb{E} \end{cases}$$

We start with a spec of the procedure ISEMIRP.

$$\textbf{proc}\ \textsc{isemirp}(\textbf{result}\ x : \mathbb{N}, \textbf{value}\ y : \mathbb{N}) \cdot$$

$$\llcorner x : [\ y > 0,\ x = f(y)\ ]\lrcorner_{(1)}$$

(1) $\sqsubseteq$ $\langle$**i-loc**$\rangle$

$$\llcorner \textbf{var}\ r\ \cdot\ x, r : [\ y > 0,\ x = f(y)\ ]\lrcorner_{(2)}$$

(2) $\sqsubseteq$ $\langle$**seq2**$\rangle$

$$\llcorner r : [\ y > 0,\ y > 0 \wedge r = 0\ ];\lrcorner_{(3)}$$

$$\llcorner x, r : [\ y > 0 \wedge r = 0,\ x = f(y)\ ]\lrcorner_{4)}$$

(3) $\sqsubseteq$ $\langle$**ass - justification 2.2.1 below**$\rangle$

$$r := 0$$

(4) $\sqsubseteq$ $\langle$**i-con**$\rangle$

$$\textbf{con}\ S : [10]^* \cdot\ x, r : \left[\ y > 0 \wedge r = 0 \wedge y = \textstyle\sum_{i=0}^{C(y)} S_i 10^{C(y)-i},\ x = f(y)\ \right]$$

$\sqsubseteq$ $\langle$**seq2**$\rangle$

$$\llcorner \textbf{con}\ S : [10]^* \cdot\ r : \left[\begin{array}{l} y > 0 \wedge r = 0 \wedge y = \sum_{i=0}^{C(y)} S_i 10^{C(y)-i}, \\ \\ y > 0 \wedge y = \sum_{i=0}^{C(y)} S_i 10^{C(y)-i} \wedge r = \sum_{i=0}^{C(y)} S_i 10^i \end{array}\right];\lrcorner_{(5)}$$

$$\llcorner \textbf{con}\ S : [10]^* \cdot\ x, r : \left[\ y > 0 \wedge y = \textstyle\sum_{i=0}^{C(y)} S_i 10^{C(y)-i} \wedge r = \sum_{i=0}^{C(y)} S_i 10^i,\ x = f(y)\ \right];\lrcorner_{(6)}$$

(5) $\sqsubseteq$ $\langle$**s-post - justification 2.2.2 below**$\rangle$

$$\textbf{con}\ S : [10]^* \cdot\ r : \left[\ y > 0 \wedge r = 0 \wedge y = \textstyle\sum_{i=0}^{C(y)} S_i 10^{C(y)-i},\ r = \sum_{i=0}^{C(y)} S_i 10^i\ \right];$$

$\sqsubseteq$ $\langle$**w-pre - justification 2.2.2 below**$\rangle$

$$\textbf{con}\ S : [10]^* \cdot\ r : \left[\ y > 0 \wedge y = \textstyle\sum_{i=0}^{C(y)} S_i 10^{C(y)-i},\ r = \sum_{i=0}^{C(y)} S_i 10^i\ \right];$$

$\sqsubseteq$ $\langle$**proc - justification 2.2.2 below**$\rangle$

$$\text{reversen}(y, r)$$

$(6) \sqsubseteq \quad \langle \textbf{i-loc} \rangle$

$$\textbf{var}\ e : \mathbb{B}\ .\ x, r, e : \left[\ y > 0 \wedge y = \sum_{i=0}^{C(y)} S_i 10^{C(y)-i} \wedge r = \sum_{i=0}^{C(y)} S_i 10^i, x = f(y)\ \right]$$

$\sqsubseteq \quad \langle \textbf{c-frame r nor r}_0 \textbf{ in post} \rangle$

$$x, e : \left[\ y > 0 \wedge y = \sum_{i=0}^{C(y)} S_i 10^{C(y)-i} \wedge r = \sum_{i=0}^{C(y)} S_i 10^i,\ x = f(y)\ \right]$$

$\sqsubseteq \quad \langle \textbf{seq and c-frame on (8) where e nor e}_0 \textbf{ is post} \rangle$

$$\llcorner e : \left[\begin{array}{l} y > 0 \wedge y = \sum_{i=0}^{C(y)} S_i 10^{C(y)-i} \wedge r = \sum_{i=0}^{C(y)} S_i 10^i, \\ y = \sum_{i=0}^{C(y)} S_i 10^{C(y)-i} \wedge r = \sum_{i=0}^{C(y)} S_i 10^i \wedge e \Leftrightarrow (y \in \mathbb{P} \wedge r \in \mathbb{P} \wedge r \neq y) \end{array} \right] ;\lrcorner(7)$$

$$\llcorner x : \left[\begin{array}{l} y = \sum_{i=0}^{C(y)} S_i 10^{C(y)-i} \wedge r = \sum_{i=0}^{C(y)} S_i 10^i \wedge e \Leftrightarrow (y \in \mathbb{P} \wedge r \in \mathbb{P} \wedge r \neq y), \\ x = f(y) \end{array} \right] \lrcorner(8)$$

$(7) \sqsubseteq \quad \langle \textbf{s-post - justification 2.2.3 below} \rangle$

$$e : \left[\begin{array}{l} y > 0 \wedge y = \sum_{i=0}^{C(y)} S_i 10^{C(y)-i} \wedge r = \sum_{i=0}^{C(y)} S_i 10^i, \\ e \Leftrightarrow (y \in \mathbb{P} \wedge r \in \mathbb{P} \wedge r \neq y) \end{array} \right];$$

$\sqsubseteq \quad \langle \textbf{w-pre - justification 2.2.3 below} \rangle$

$$e : \left[\ True,\ e \Leftrightarrow (y \in \mathbb{P} \wedge r \in \mathbb{P} \wedge r \neq y)\ \right];$$

$\sqsubseteq \quad \langle \textbf{proc - justification 2.2.3 below} \rangle$

$$e := \text{mpz\_probab\_prime\_p}(y) \wedge \text{mpz\_probab\_prime\_p}(r) \wedge \text{mpz\_cmp}(y, r)$$

$(8) \sqsubseteq \quad \langle \textbf{if} \rangle$

if e **then**

$$\llcorner x : \left[\begin{array}{l} e \Leftrightarrow \text{TRUE} \Leftrightarrow (y \in \mathbb{P} \wedge r \in \mathbb{P} \wedge r \neq y) \wedge y = \sum_{i=0}^{C(y)} S_i 10^{C(y)-i} \wedge r = \sum_{i=0}^{C(y)} S_i 10^i, \\ x = f(y) \end{array} \right] \lrcorner(9)$$

else

$$\llcorner x : \left[\begin{array}{l} e \Leftrightarrow \text{FALSE} \Leftrightarrow (y \in \mathbb{P} \wedge r \in \mathbb{P} \wedge r \neq y) \wedge y = \sum_{i=0}^{C(y)} S_i 10^{C(y)-i} \wedge r = \sum_{i=0}^{C(y)} S_i 10^i, \\ x = f(y) \end{array} \right] \lrcorner(10)$$

$(9) \sqsubseteq \quad \langle \textbf{ass - justfication 2.2.4 below} \rangle$

$$x := 1$$

$(10) \sqsubseteq \quad \langle \textbf{ass - justfication 2.2.5 below} \rangle$

$$x := 0$$

There are still some outstanding proofs in the above derivation. We will discharge them below in the next subsection.

2.2.1 Proof of $(3) \sqsubseteq r := 0$

We need to prove the validity of

$$pre(3) \Rightarrow (post(3))[^0/_r]$$

Proving from the bottom up, we have:

$$
\begin{array}{ll}
 & pre(3) \\
\Leftrightarrow & \quad \langle \text{Definition} \rangle \\
 & y > 0 \\
\Leftrightarrow & \quad \langle \text{Second conjunct below is logically true. A} \wedge \text{T is equivalent to A} \rangle \\
 & y > 0 \wedge 0 = 0 \\
\Leftrightarrow & \quad \langle \text{Definition and substituition} \rangle \\
 & post(3)[^0/_r]
\end{array}
$$

2.2.2 Proof of $(5) \sqsubseteq$ reversen(y,r)

We use the definition of REVERSEN from the Assignment 2 Specification which reads:

$$
\begin{array}{l}
\textbf{proc } \text{REVERSEN}(\textbf{value } n : \mathbb{N}, \textbf{result } r : \mathbb{N}) \cdot \\
\quad \textbf{con } S : [10]^* \ . \ r : \left[\ n = \sum_{i=0}^{c(n)} (S_i 10^{(c(n)-i)}) \wedge n > 0, r = \sum_{i=0}^{c(n)} (S_i 10^i) \ \right]
\end{array}
$$

To prove this that (5) can be refined into the required procedure, we must follow the w-pre and s-post rules as described in the COMP2111 glossary. Here we use y as the value parameter in place of n.

We start be strengthening the postcondition. We must prove:

$$pre(5)[^{r_0}/_r] \wedge post(\text{REVERSEN}(y, r)) \Rightarrow post(5)$$

Proving from the top down:

$$
\begin{array}{ll}
 & pre(5)[^{r_0}/_r] \wedge post(\text{REVERSEN}(y, r)) \\
\Leftrightarrow & \quad \langle \text{Substitution} \rangle \\
 & y > 0 \wedge r_0 = 0 \wedge y = \sum\limits_{i=0}^{C(y)} S_i 10^{C(y)-i} \wedge r = \sum\limits_{i=0}^{c(n)} (S_i 10^i) \\
\Rightarrow & \quad \langle \text{Discarding the second conjunct as A} \wedge \text{B} \Rightarrow \text{B} \rangle \\
 & y > 0 \wedge y = \sum\limits_{i=0}^{C(y)} S_i 10^{C(y)-i} \wedge r = \sum\limits_{i=0}^{C(y)} S_i 10^i \\
\Leftrightarrow & \quad \langle \text{Definition} \rangle
\end{array}
$$

$post(5)$

Now to prove our weakening of the precondition as seen above in the derivation. We must prove:

$$pre(5) \Rightarrow pre(\textsc{reversen}(y, r))$$

Proving from the top down:

$$\begin{array}{ll} & pre(5) \\ \Leftrightarrow & \quad \langle \text{Substitution} \rangle \\ & y > 0 \wedge r = 0 \wedge y = \sum_{i=0}^{c(y)} (S_i 10^{(c(y)-i)}) \\ \Rightarrow & \quad \langle \text{Discard second conjunct as A} \wedge \text{B} \Rightarrow \text{A} \rangle \\ & y = \sum_{i=0}^{c(y)} (S_i 10^{(c(y)-i)}) \wedge y > 0 \\ \Leftrightarrow & \quad \langle \text{Substitution, noting we are using } y \text{ as input} \rangle \\ & pre(\textsc{reversen}(y, r)) \end{array}$$

Therefore, by w-pre and s-post, we have refined down to the specifications of the procedure allowing a valid call to it.

2.2.3 Proof of $(7) \sqsubseteq e :=$ mpz-probab-prime-p(y) $\wedge$ mpz-probab-prime-p(r) $\wedge$ mpz-cmp(y,r)

We specify MPZ-PROBAB-PRIME-P and MPZ-CMP as:

$$\begin{array}{l} \textbf{proc } \textsc{mpz-probab-prime-p}(\textbf{value } x : \mathbb{N}, \textbf{result } y : \mathbb{B}) \cdot \\ \quad y : \left[\text{ TRUE}, x \Leftrightarrow (y \in \mathbb{P}) \right] \end{array}$$

$$\begin{array}{l} \textbf{proc } \textsc{mpz-cmp}(\textbf{value } y : \mathbb{Z}, \textbf{value } z : \mathbb{Z}, \textbf{result } x : \mathbb{B}) \cdot \\ \quad y : \left[\text{ TRUE}, x \Leftrightarrow (y = z) \right] \end{array}$$

Similarly to 2.2.2, we begin my strengthening the postcondition. We need to prove: $\text{pre}(7)[^{e_0}/_e] \wedge e \Leftrightarrow (y \in \mathbb{P} \wedge r \in \mathbb{P} \wedge r \neq y) \Rightarrow \text{post}(7)$

$$\begin{array}{ll} & pre(7)[^{e_0}/_e] \wedge e \Leftrightarrow (y \in \mathbb{P} \wedge r \in \mathbb{P} \wedge r \neq y) \\ \Leftrightarrow & \quad \langle \text{Definition and substituition} \rangle \end{array}$$

$$
\begin{aligned}
& y > 0 \land y = \sum_{i=0}^{C(y)} S_i 10^{C(y)-i} \land r = \sum_{i=0}^{C(y)} S_i 10^i \land e \Leftrightarrow (y \in \mathbb{P} \land r \in \mathbb{P} \land r \neq y) \\
\Rightarrow \quad & \langle \text{Discarding the first conjunct} \rangle \\
& y = \sum_{i=0}^{C(y)} S_i 10^{C(y)-i} \land r = \sum_{i=0}^{C(y)} S_i 10^i \land e \Leftrightarrow (y \in \mathbb{P} \land r \in \mathbb{P} \land r \neq y)
\end{aligned}
$$

Now we must weaken the precondition as to refine to our procedure call. To do such we must prove:
$y > 0 \land y = \sum_{i=0}^{C(y)} S_i 10^{C(y)-i} \land r = \sum_{i=0}^{C(y)} S_i 10^i \Rightarrow True$

As the precondition is taken to be true. True can of course be implied from True.

$$
\begin{aligned}
& y > 0 \land y = \sum_{i=0}^{C(y)} S_i 10^{C(y)-i} \land r = \sum_{i=0}^{C(y)} S_i 10^i \\
\Rightarrow \quad & \langle \text{Using the fact that A} \land \text{T} \Rightarrow \text{T} \rangle \\
& \text{True}
\end{aligned}
$$

By s-post and w-pre rules defined by the COMP2111 Glossary and expanding the post-condition, we now have:

$$
e : [\text{TRUE},\ e \Leftrightarrow (y \in \mathbb{P}) \land e \Leftrightarrow (r \in \mathbb{P}) \land e \Leftrightarrow (r \neq y)]
$$

This meets the function specifications of mpz_probab_prime_p and mpz_cmp as defined above. Note that we are taking the intersection of all three calls simultaneously to forego writing each function refinement seperately.

2.2.4 Proof of $(9) \sqsubseteq x := 1$

We need to prove the validity of

$$
pre(9) \Rightarrow (post(9))[^1/_x]
$$

i.e., the prerequisite of the relevant instance of **ass**. Expanding the definitions and performing the substitution yields

$$
\left(y = \sum_{i=0}^{C(y)} S_i 10^{C(y)-i} \land r = \sum_{i=0}^{C(y)} S_i 10^i \land e \Leftrightarrow \text{TRUE} \Leftrightarrow (y \in \mathbb{P} \land r \in \mathbb{P} \land r \neq y) \right) \Rightarrow
$$
$$
f(y) = 1 \ .
$$

The items in *blue* and *red* form the first two conjuncts of our definition of emirps $\mathbb{E}$ (see Task 1). The equivalence in *orange* shows that $y \in \mathbb{P} \land r \in \mathbb{P} \land x \neq r$ are all evaluated true in our pre-condition. These three conditions form the needed three conjuncts for our definition of $\mathbb{E}$. Therefore, $y \in \mathbb{E}$. We then notice that because $y \in \mathbb{E}$ and by our definition of $f(y)$ (see Task 1), $f(y) = 1$, resulting in the postcondition.

2.2.5 Proof of $(10) \sqsubseteq x := 0$

We need to prove the validity of

$$pre(10) \Rightarrow (post(10))[^0/_x]$$

i.e., the prerequisite of the relevant instance of **ass**. Expanding the definitions and performing the substitution yields

$$\left(y = \sum_{i=0}^{C(y)} S_i 10^{C(y)-i} \wedge r = \sum_{i=0}^{C(y)} S_i 10^i \wedge e \Leftrightarrow \text{FALSE} \Leftrightarrow (y \in \mathbb{P} \wedge r \in \mathbb{P} \wedge r \neq y) \right) \Rightarrow$$
$$f(y) = 0 \ .$$

The items in *blue* and *red* form the first two conjuncts of our definition of emirps $\mathbb{E}$ (see Task 1). The equivalence in *orange* shows that $y \in \mathbb{P} \wedge r \in \mathbb{P} \wedge x \neq r$ are collectively evaluated false in our pre-condition. This means one of these conditions is false. Looking at our definition of $\mathbb{E}$, the variable y is not contained within the set $\mathbb{E}$ for this reason. We then notice that because $y \notin \mathbb{E}$ and by our definition of $f(y)$ (see Task 1), $f(y) = 0$, resulting in our postcondition.

3 Task 3

```
#include <stdio.h>
#include <stdlib.h>
#include <gmp.h>
#include <reverse.h>

int is_emirp(mpz_t z);
void emirp(mpz_t n, unsigned long emirpNumber);

int is_emirp(mpz_t y)
{
        mpz_t r;        // Declares a gmp number called r
        mpz_init(r);        // A gmp function call to initilize r and set it to 0
        reversen(y, r);        // Function call to set the value or r to the reverse value of n
        if ((mpz_cmp(y, r)) &&        // Check for emirp by emirp definition
                (mpz_probab_prime_p(y, 5)) &&
                (mpz_probab_prime_p(r, 5))) {
                return 1;
        }
        return 0;
}

void emirp(mpz_t n, unsigned long emirpNumber)
{
        // Set up the loop invariant
        mpz_init(n);        // A gmp function call to initialize n and set it to 0
        int emirpsCounted = 0;        // The number of emirps the program has counted so far
        int m = is_emirp(n);        // This is required for the loop invariant

        // Run the loop
        while (emirpsCounted != emirpNumber || m != 1) {
                mpz_add_ui(n, n, 1);        // Add unsigned long value '1' to mpz_t 'n'
                m = is_emirp(n);
                if (m == 1) {
                        emirpsCounted += 1;
                }
        }
}

int main(void)
{
        int emirpNumber = 0;        // Get user input to get what number emirp is desired
```

```c
	printf("Enter a number: ");
	if (!scanf("%d", &emirpNumber)) {
		printf("Invalid input\n");
		exit(1);
	}

	mpz_t n;	// Declares a gmp number called n
	mpz_init(n);	// A gmp function call to initialize n and set it to 0
	emirp(n, emirpNumber);	// Set n to the value of the emirpNumber'th emirp
	gmp_printf("%Zd\n", n);	// Prints the emirpNumber'th emirp

	return 0;
}
```

In the C code we decided to do so and so