Assignment 2

Ruofei HUANG(z5141448) Anqi ZHU(z5141541)

April 28, 2018

1 Task 1

1.1 Prime

A number is a prime is a nautal number greater than 1 that cannot be formed by mutiplying two smaller number[1]. Hence we can have the definition by set theory:

$$Prime = \{n | \neg\exists x \in (2..n-1)\,(x|n)\}$$

Then we can base on the GMP function that make up the spec of the procedure call of **proc** $ISPRIME(\textbf{value}\ n, \textbf{result}\ r)$

$$\textbf{proc}\ ISPRIME(\textbf{value}\ n, \textbf{result}\ r)\cdot$$
$$[TRUE, (n \in Prime \land r > 0) \lor (n \notin Prime \land r = 0)]$$

Which means $ISPRIME(n) > 0$[2] $\Rightarrow n \in Prime$.

1.2 Reverse

To define a emirp, a mathmatical function to decribe a number's reverse would be helpful. By the spec in verifying reversen[3], we have this definition:

$$rev(n) = \sum_{i=0}^{c(n)} (S_i 10^i)$$

[1] Direct reference from Wikipedia: https://en.wikipedia.org/wiki/Prime_number

[2] This act as another procedure call sugar $r := ISPRIME(n)$; the complete expanded version will be $ISPRIME(n,r); ((r > 0 \Rightarrow r \in Prime) \lor (\textbf{else} \Rightarrow r \notin Prime))$. The continueous prove will follow this convension to make life easier.

[3] A proof provided by Lecturer in Control of this course on https://www.cse.unsw.edu.au/~cs2111/18s1/lec/reverse.pdf

where:

$$c(n) = \lfloor log_{10}(n) \rfloor,$$
$$S = [10]^*,$$
$$n \in \mathbb{N} \land n = \sum_{i=0}^{c(n)} (S_i 10^{(c(n)-i)})$$

Hence, we can justify the spec of $\mathbf{proc}\ reversen(\mathbf{value}\ n : \mathbb{Z}, \mathbf{result}\ r : \mathbb{Z})$ as

$$\mathbf{proc}\ reversen(\mathbf{value}\ n : \mathbb{Z}, \mathbf{result}\ r : \mathbb{Z})\cdot$$
$$r : [\text{TRUE}, r = rev(n)]$$

1.3 Emirp

An emirp is a prime number that results iin a different prime when its decimal digits are reverse[4]. Hence a definition of emirp can be construct as follow:

$$n \in Emirp \iff n \in Prime \land rev(n) \in Prime$$

We also construct another function to help us find the n^{th} emirp, which is as follow:

$$isEmirp(n) = \begin{cases} 0 & \text{if } n \in Prime \land rev(n) \in Prime \\ 1 & \text{else} \end{cases}$$

1.3.1 Procedure Call

Also for our usage in the procedure call in the main programme, we have develop a procedure to do the same thing. Hence we have this spec

$$\mathbf{proc}\ ISEMIRP(\mathbf{value}\ n : \mathbb{N}, \mathbf{result}\ w)\cdot$$
$$\llcorner n, w : [\text{TRUE}, \begin{pmatrix} (w = 1 \land rev(n) \neq n \land n \in Prime \land rev(n) \in Prime) \lor \\ (w \neq 1 \land \neg(rev(n) \neq n \land n \in Prime \land rev(n) \in Prime)) \end{pmatrix}]\lrcorner_{(A)}$$

1.3.2 Refinement Calculatiion

$$(A) \sqsubseteq \quad \langle \textbf{c-frame} \rangle$$
$$w : [\text{TRUE}, \begin{pmatrix} (w = 1 \land rev(n) \neq n \land n \in Prime \land rev(n) \in Prime) \lor \\ (w \neq 1 \land \neg(rev(n) \neq n \land n \in Prime \land rev(n) \in Prime)) \end{pmatrix}]$$

[4] Another reference from Wikipedia :https://en.wikipedia.org/wiki/Emirp

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

$r \cdot w : [\text{TRUE}, \begin{pmatrix} (w = 1 \land rev(n) \neq n \land n \in Prime \land rev(n) \in Prime) \lor \\ (w \neq 1 \land \neg(rev(n) \neq n \land n \in Prime \land rev(n) \in Prime)) \end{pmatrix}]$

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

$\llcorner r : [\text{TRUE}, r = rev(n)];\lrcorner_{(B)}$

$\llcorner r \cdot w : [r = rev(n), \begin{pmatrix} (w = 1 \land rev(n) \neq n \land n \in Prime \land rev(n) \in Prime) \lor \\ (w \neq 1 \land \neg(rev(n) \neq n \land n \in Prime \land rev(n) \in Prime)) \end{pmatrix}$

$(B) \sqsubseteq$ $\langle$**ass**,need to be justify? $\rangle$

$r := reversen(n)$

$(C) \sqsubseteq$ $\langle$**s-pos**, $r = rev(n)$, replace all the rev in the post$\rangle$

$\llcorner r \cdot w : [r = rev(n), \begin{pmatrix} (w = 1 \land r \neq n \land n \in Prime \land r \in Prime) \lor \\ (w \neq 1 \land \neg(r \neq n \land n \in Prime \land r \in Prime)) \end{pmatrix}]\lrcorner_{(C')}$

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

if $r \neq n$

then$\llcorner r \cdot w : [r \neq n, post(C')]\lrcorner_{(D)}$

else$\llcorner r \cdot w : [r = n, post(C')]\lrcorner_{(E)}$

fi

$(D) \sqsubseteq$ $\langle$**if**$\rangle$

if $ISPRIME(n) > 0$

then$\llcorner r \cdot w : [r \neq n \land n \in Prime, post(C')]\lrcorner_{(F)}$

else$\llcorner r \cdot w : [r \neq n \land n \notin Prime, post(C')]\lrcorner_{(G)}$

fi

$(F) \sqsubseteq$ $\langle$**if**$\rangle$

if $ISPRIME(r) > 0$

then$\llcorner r \cdot w : [r \neq n \land n \in Prime \land r \in Prime, post(C')]\lrcorner_{(G)}$

else$\llcorner r \cdot w : [r \neq n \land n \in Prime \land r \notin Prime, post(C')]\lrcorner_{(H)}$

fi

$(E)(G)(H) \sqsubseteq$ $\langle$**ass**, need to be justify$\rangle$

$w := 0$

$(G) \sqsubseteq$ $\langle$**ass**, need to be justify$\rangle$

$w := 1$

1.3.3 Justification of

1.3.4 Toy Language Code

$r := reversen(n)$**if** $r \neq n$
then
  if $ISPRIME(n) > 0$
  then
    if $ISPRIME(r) > 0$
    then $w := 1$
    else $w := 0$
    fi
  else $w := 0$
  fi
else $w := 0$
fi

1.4 Pre- and Postcondition

Our task is find the n^{th} emprip, by the previous definition of $isEmirp(n)$ we can construct the pre- and postcondition in this way:

proc $EMIRP($**value** $n : \mathbb{N},$ **result** $r)$

$$\llcorner n : \mathbb{N}, \textbf{result } r : \mathbb{N}[\textsc{true}, \sum_{i=0}^{r} isEmirp(i) = n \land r \in Emirp]\lrcorner_{(1)}$$

2 Task 2

$$(1) \sqsubseteq \quad \langle \textbf{c-frame} \rangle$$

$$r[\text{TRUE}, \sum_{i=0}^{r} isEmirp(i) = n \wedge r \in Emirp]$$

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

$$count \cdot r[\text{TRUE}, \sum_{i=0}^{r} isEmirp(i) = n \wedge r \in Emirp]$$

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

$$\llcorner count, r[\text{TRUE}, Inv]\lrcorner_{(2)};$$

$$\llcorner count, r[Inv, \sum_{i=0}^{r} isEmirp(i) = n \wedge r \in Emirp]\lrcorner_{(3)}$$

Where the loop invariant is defined by:

$$Inv = \left(\; count = \sum_{i=0}^{count} isEmirp(i) \wedge count \leq n \; \right)$$

$$(2) \sqsubseteq \quad \langle \text{Routine work, set up the variable of loop} \rangle$$

$$r := 1; count := 0;$$

$$(3) \sqsubseteq \quad \langle \textbf{s-post} \rangle$$

$$count, r[Inv, Inv \wedge count = n]$$

$$\sqsubseteq \quad \langle \textbf{while} \rangle$$

while $count \neq n$ **do**

$$\llcorner count, r[Inv \wedge count \neq n, Inv]\lrcorner_{4}$$

od

$$(4) \sqsubseteq \quad \langle \textbf{seq2} \rangle$$

$$\llcorner r[Inv \wedge count \neq n, Inv[^{r+1}/_{r}]];\lrcorner_{(5)}$$

$$\llcorner count, r[Inv[^{r+1}/_{r}], Inv]\lrcorner_{(6)}$$

$$(5) \sqsubseteq \quad \langle \textbf{ass, need to justify?} \rangle$$

$$r := r + 1;$$

$$(6) \sqsubseteq \quad \langle \textbf{ass, need to justify?} \rangle$$

$$count := count + ISEMIRP(r);$$

2.0.1 Toy Language Programme

We collect all out toy language code, we have

$$
\begin{array}{l}
r := 1; count := 0; \\
\textbf{while } count \neq n \textbf{ do} \\
\quad r := r + 1; \\
\quad count := count + ISEMIRP(r); \textbf{od}
\end{array}
$$

2.1 Task 3

2.2 Task 4