

Assignment 2 - Emirps

Rahil Agrawal z5165505

Aditya Karia z5163287

COMP2111 18s1

1 Task 1 - Specification Statement

A prime number is a positive integer that is only divisible by 1 and itself. Therefore, we can say that a number r is prime if it is not divisible by any number between 2 and $r - 1$ inclusive.

Therefore, we can define a primality check function as follows:

$$isPrime(r) = \begin{cases} \text{true} & \neg \exists k \in 2..(r-1) (r \bmod k = 0) \\ \text{false} & \exists k \in 2..(r-1) (r \bmod k = 0) \end{cases}$$

The $reverse(r, s)$ function can be used to store the reverse of a number r in a variable called s .

Having defined a primality check function $isPrime(r, a)$ and a function to store the reverse of a number r in s , we define an *emirp*.

An *emirp* is a prime number whose reversal is also prime, but which is not a palindromic prime.

Therefore, if $EMIRP(r, n)$ states that r is the n^{th} *emirp*, where n is a positive integer, then:

$$EMIRP(r, n) = \begin{cases} \text{true} & isPrime(r) \wedge reverse(r, s) \wedge isPrime(s) \wedge r \neq s \\ \text{false} & otherwise \end{cases}$$

We want to find the n^{th} *emirp*. Having defined the limitations on n ($n > 0$) and described what it means for a r to be the n^{th} , we can specify our program with by:

proc EMIRP(**value** n , **result** r) $\cdot \sqcup n, r, x : [n > 0, Emirp(r, n)] \dashv(1)$

2 Task 2 - Derivation

$$\begin{aligned}
& \text{proc EMIRP}(\text{value } n, \text{result } r) \cdot \sqcup n, r, x : [n > 0, \text{Emirp}(r, n)] \text{ } \textcolor{red}{\dashv(1)} \\
(1) \sqsubseteq & \quad \langle \text{c-frame} \rangle \\
& \sqcup r, x : [n > 0, \text{Emirp}(r, n)] \text{ } \textcolor{red}{\dashv(2)} \\
(2) \sqsubseteq & \quad \langle \text{i-loc} \rangle \\
& \sqcup i, r, x : [n > 0, \text{Emirp}(r, n)] \text{ } \textcolor{red}{\dashv(3)} \\
(3) \sqsubseteq & \quad \langle \text{seq} \rangle \\
& \sqcup i, x, r : [n > 0, i = 1 \wedge x = 13 \wedge n > 0] \text{ } \textcolor{red}{\dashv(4)}; \\
& \sqcup i, x : [i = 1 \wedge x = 13 \wedge n > 0, \text{Emirp}(r, n)] \text{ } \textcolor{red}{\dashv(5)} \\
(4) \sqsubseteq & \quad \langle \text{c-frame} \rangle \\
& i, x : [n > 0, i = 1 \wedge x = 13 \wedge n > 0] \\
& \sqsubseteq \quad \langle \text{ass - (1)} \rangle \\
& \textcolor{blue}{i} := \textcolor{blue}{1} \\
& \textcolor{blue}{x} := \textcolor{blue}{13} \\
(5) \sqsubseteq & \quad \langle \text{seq} \rangle \\
& \sqcup i, r, x : [i = 1 \wedge x = 13 \wedge n > 0, \text{Inv}] \text{ } \textcolor{red}{\dashv(6)}; \\
& \sqcup i, r, x : [\text{Inv}, \text{Inv} \wedge i = n] \text{ } \textcolor{red}{\dashv(7)}; \\
& \sqcup i, r, x : [\text{Inv} \wedge i = n, \text{Emirp}(r, n)] \text{ } \textcolor{red}{\dashv(8)} \\
(6) \sqsubseteq & \quad \langle \text{w-pre, c-frame - (2)} \rangle \\
& r, x : [\text{Inv}^{[13/r]}, \text{Inv}] \\
& \sqsubseteq \quad \langle \text{ass - (3)} \rangle \\
& \textcolor{blue}{r} := \textcolor{blue}{13} \\
(7) \sqsubseteq & \quad \langle \text{while} \rangle \\
& \textcolor{blue}{while } i \neq n \textcolor{blue}{ do} \\
& \quad \sqcup i, r, x : [\text{Inv} \wedge i \neq n, \text{Inv}] \text{ } \textcolor{red}{\dashv(9)} \\
& \textcolor{blue}{od;} \\
(8) \sqsubseteq & \quad \langle \text{w-pre, c-frame - (4)} \rangle \\
& \sqcup r, x : [\text{EMIRP}(r, n), \text{EMIRP}(r, n)] \text{ } \textcolor{red}{\dashv(10)} \\
& \sqsubseteq \quad \langle \text{skip - (5)} \rangle \\
& \textcolor{blue}{skip} \\
(9) \sqsubseteq & \quad \langle \text{seq} \rangle \\
& \sqcup r, x : [\text{Inv} \wedge i \neq n, \text{Inv}^{[x+1/x]}] \text{ } \textcolor{red}{\dashv(10)}; \\
& \sqcup r, x : [\text{Inv}^{[x+1/x]}, \text{Inv}] \text{ } \textcolor{red}{\dashv(11)}
\end{aligned}$$

$$\begin{aligned}
(10) &\sqsubseteq \langle \text{ass - (6)} \rangle \\
&\quad \mathbf{x} := \mathbf{x} + 1 \\
(11) &\sqsubseteq \langle \text{i-loc, seq} \rangle \\
&\quad \sqsubseteq a, i, r, x : [\text{Inv}^{[x+1/x]}, \text{Inv}^{[x+1/x]} \wedge a = 1] \neg(12); \\
&\quad \sqsubseteq a, i, r, x : [\text{Inv}^{[x+1/x]} \wedge a = 1, \text{Inv}] \neg(13) \\
(12) &\sqsubseteq \langle \text{c-frame} \rangle \\
&\quad a, x : [\text{Inv}^{[x+1/x]}, \text{Inv}^{[x+1/x]} \wedge a = 1] \\
&\sqsubseteq \langle \text{ass - (7)} \rangle \\
&\quad \mathbf{a} := 1 \\
(13) &\sqsubseteq \langle \text{seq} \rangle \\
&\quad \sqsubseteq a, i, r, x : \left[\begin{array}{l} \text{Inv}^{[x+1/x]} \wedge a = 1, (a = 1 \wedge \neg \exists k \in 2..(x-1) (x \bmod k = 0)) \\ \vee (a = 0 \wedge \exists k \in 2..(x-1) (x \bmod k = 0)) \end{array} \right] \neg(14); \\
&\quad \sqsubseteq a, i, r, x : \left[\begin{array}{l} (a = 1 \wedge \neg \exists k \in 2..(x-1) (x \bmod k = 0)) \\ \vee (a = 0 \wedge \exists k \in 2..(x-1) (x \bmod k = 0)), \text{Inv} \end{array} \right] \neg(15) \\
(14) &\sqsubseteq \langle \text{w-pre - (8)} \rangle \\
&\quad a, x : [a = 1 \wedge x > 0, \text{post}(14)] \\
&\sqsubseteq \langle \text{proc} \rangle \\
&\quad \text{isPrime}(\mathbf{x}, \mathbf{a}) \\
(15) &\sqsubseteq \langle \text{if} \rangle \\
&\quad \text{if } \mathbf{a} = 1 \\
&\quad \text{then } \sqsubseteq a, i, r, x : [a = 1 \wedge \text{pre}(15), \text{post}(15)] \neg(16) \\
&\quad \text{else } \sqsubseteq p, x : [a \neq 1 \wedge \text{pre}(15), \text{post}(15)] \neg(17) \\
&\quad \text{fi} \\
(16) &\sqsubseteq \langle \text{i-loc} \rangle \\
&\quad a, i, r, s, x : [\text{pre}(16), \text{post}(16)] \\
&\sqsubseteq \langle \text{seq} \rangle \\
&\quad \sqsubseteq s, x : [\text{pre}(16), s = 0] \neg(18); \\
&\quad \sqsubseteq a, i, r, s, x : [s = 0, \text{post}(16)] \neg(19) \\
(17) &\sqsubseteq \langle \text{c-frame, w-pre - (9)} \rangle \\
&\quad i, r : [\text{Inv}, \text{Inv}] \\
&\sqsubseteq \langle \text{skip - (10)} \rangle \\
&\quad \text{skip} \\
(18) &\sqsubseteq \langle \text{ass - (11)} \rangle \\
&\quad \mathbf{s} := 0 \\
(19) &\sqsubseteq \langle \text{seq} \rangle \\
&\quad \sqsubseteq s, x : [\text{pre}(19), \text{reversen function post condition}] \neg(20); \\
&\quad \sqsubseteq a, i, r, s, x : [\text{reversen function post condition}, \text{post}(19)] \neg(21)
\end{aligned}$$

(20) \sqsubseteq $\langle \text{i-con, c-frame, w-pre - (12)} \rangle$
 $\text{con } S : [10]^* \cdot s : [\text{reverse function pre condition, reversen function post condition}]$
 \sqsubseteq $\langle \text{proc} \rangle$
 $\text{reversen}(\mathbf{x}, \mathbf{s})$

(21) \sqsubseteq $\langle \text{i-loc, seq} \rangle$
 $\sqsubseteq a, i, r, s, b, x : [\text{pre}(21), \text{pre}(21) \wedge b = 1] \neg(22);$
 $\sqsubseteq a, i, r, s, b, x : [\text{pre}(21) \wedge b = 1, \text{post}(21)] \neg(23)$

(22) \sqsubseteq $\langle \text{c-frame, ass - (13)} \rangle$
 $\mathbf{b} := 1$

(23) \sqsubseteq $\langle \text{seq} \rangle$
 $\sqsubseteq a, i, r, s, b, x : \left[\begin{array}{l} \text{pre}(21) \wedge b = 1, (b = 1 \wedge \neg \exists k \in 2..(s-1) (s \bmod k = 0)) \\ \vee (b = 0 \wedge \exists k \in 2..(s-1) (s \bmod k = 0)) \end{array} \right] \neg(24);$
 $\sqsubseteq a, i, r, s, b, x : \left[\begin{array}{l} (b = 1 \wedge \neg \exists k \in 2..(s-1) (s \bmod k = 0)) \\ \vee (b = 0 \wedge \exists k \in 2..(s-1) (s \bmod k = 0)), \text{post}(21) \end{array} \right] \neg(25)$

(24) \sqsubseteq $\langle \text{w-pre - (14)} \rangle$
 $a, i, r, s, b, x : \left[\begin{array}{l} s > 0 \wedge b = 1, (b = 1 \wedge \neg \exists k \in 2..(s-1) (s \bmod k = 0)) \\ \vee (b = 0 \wedge \exists k \in 2..(s-1) (s \bmod k = 0)) \end{array} \right]$
 \sqsubseteq $\langle \text{proc} \rangle$
 $\text{isPrime}(\mathbf{s}, \mathbf{b})$

(25) \sqsubseteq $\langle \text{if} \rangle$
 $\text{if } \mathbf{b} = 1 \wedge \mathbf{s} \neq \mathbf{x}$
 $\text{then } \sqsubseteq i, x : [b = 1 \wedge s \neq x \wedge \text{pre}(25), \text{post}(25)] \neg(26)$
 $\text{else } \sqsubseteq i, r, a, s, b, x : [(b \neq 1 \vee s = x) \wedge \text{pre}(25), \text{post}(25)] \neg(27)$
 $\text{fi};$

(26) \sqsubseteq $\langle \text{c-frame, w-pre- (15)} \rangle$
 $a, i, r, s, b, x : [\text{Inv}[^{i+1}/i][^x/r], \text{Inv}]$
 \sqsubseteq $\langle \text{ass - (16)} \rangle$
 $\mathbf{i} := \mathbf{i} + 1$
 $\mathbf{r} := \mathbf{x}$

(27) \sqsubseteq $\langle \text{c-frame, w-pre- (17)} \rangle$
 $a, i, r, s, b, x : [\text{Inv}, \text{Inv}]$
 \sqsubseteq $\langle \text{skip - (18)} \rangle$
 skip

```

proc ISPRIME(value  $r$ , result  $a$ ) ·
     $\sqsubseteq r, a : \left[ \begin{array}{l} a = 1 \wedge r > 0, (a = 1 \wedge \neg \exists k \in 2..(r-1) (r \bmod k = 0)) \\ \vee (a = 0 \wedge \exists k \in 2..(r-1) (r \bmod k = 0)) \end{array} \right] \neg(1)$ 
(1)  $\sqsubseteq$      $\langle \text{seq, i-loc} \rangle$ 
     $\sqsubseteq r, a, j : \left[ a = 1 \wedge r > 0, a = 1 \wedge r > 0 \wedge j = 2 \right] \neg(2);$ 
     $\sqsubseteq r, a, j : \left[ \begin{array}{l} a = 1 \wedge r > 0 \wedge j = 2, (a = 1 \wedge \neg \exists k \in 2..(r-1) (r \bmod k = 0)) \\ \vee (a = 0 \wedge \exists k \in 2..(r-1) (r \bmod k = 0)) \end{array} \right] \neg(3)$ 
(2)  $\sqsubseteq$      $\langle \text{ass - (19)} \rangle$ 
     $j := 2$ 
(3)  $\sqsubseteq$      $\langle \text{seq} \rangle$ 
     $\sqsubseteq r, a, j : \left[ a = 1 \wedge r > 0 \wedge j = 2, Inv_2 \right] \neg(4);$ 
     $\sqsubseteq r, a, j : \left[ Inv_2, Inv_2 \wedge j = r \right] \neg(5);$ 
     $\sqsubseteq r, a, j : \left[ Inv_2 \wedge j = r, post(3) \right] \neg(6)$ 
(4)  $\sqsubseteq$      $\langle \text{w-pre - (20)} \rangle$ 
     $r, a, j : \left[ Inv_2, Inv_2 \right]$ 
     $\sqsubseteq$      $\langle \text{skip - (21)} \rangle$ 
     $\text{skip}$ 
(6)  $\sqsubseteq$      $\langle \text{w-pre - (22)} \rangle$ 
     $\sqsubseteq r, a, j : \left[ post(3), post(3) \right] \neg(7)$ 
     $\sqsubseteq$      $\langle \text{skip - (23)} \rangle$ 
     $\text{skip}$ 
(5)  $\sqsubseteq$      $\langle \text{while} \rangle$ 
    while  $j \neq r$  do
         $\sqsubseteq r, j : \left[ Inv_2 \wedge j \neq r, Inv_2 \right] \neg(8)$ 
    od;
(8)  $\sqsubseteq$      $\langle \text{seq} \rangle$ 
     $\sqsubseteq r, j : \left[ pre(8), Inv_2[j^{+1}/j] \right] \neg(9);$ 
     $\sqsubseteq r, j : \left[ Inv_2[j^{+1}/j], Inv_2 \right] \neg(10)$ 
(9)  $\sqsubseteq$      $\langle \text{if} \rangle$ 
    if  $r \bmod j = 0$ 
    then  $\sqsubseteq a : \left[ r \bmod j = 0 \wedge pre(9), post(9) \right] \neg(11)$ 
    else  $\sqsubseteq a : \left[ r \bmod j \neq 0 \wedge pre(9), post(9) \right] \neg(12)$ 
    fi;
(10)  $\sqsubseteq$      $\langle \text{ass - (24)} \rangle$ 
     $j := j + 1$ 

```

$$\begin{aligned}
(11) &\sqsubseteq \langle \text{w-pre - (25)} \rangle \\
&\quad r, a, j : [\text{Inv}_2^{[j+1/j]}[{}^0/a], \text{post}(11)] \\
&\sqsubseteq \langle \text{ass - (26)} \rangle \\
&\quad \mathbf{a} := 0 \\
(12) &\sqsubseteq \langle \text{w-pre - (27)} \rangle \\
&\quad r, a, j : [\text{Inv}_2^{[j+1/j]}, \text{post}(11)] \\
&\sqsubseteq \langle \text{skip - (28)} \rangle \\
&\quad \text{skip}
\end{aligned}$$

We gather the code for the procedure body of EMIRP:

```

EMIRP(r, n) :
  var i := 1;
  var x := 13;
  r := 13;
  while j ≠ r do
    x := x + 1;
    var a := 1;
    isPrime(x, a);
    if a = 1 then
      var s := 0;
      reversen(x, s);
      var b := 1;
      isPrime(s, b);
      if b = 1 ∧ s ≠ x then
        i := i + 1;
        r := x;
  od;

```

Also, we gather the code for the procedure body of ISPRIME:

```

isPrime(r,j) :
  var j := 2;
  while j ≠ r do
    if (r mod j) = 0 then
      a := 0;
      j := j + 1;
  od;

```

We have derived our code. However we need to prove **some** refinements.

2.1 Implication 1: $(4) \sqsubseteq i := 1$

To prove: $i = i_0 \wedge n > 0 \Rightarrow (i = 1 \wedge x = 13 \wedge n > 0)^{[1/i]}[^{13/x}]$

Proof:

$LHS = i = i_0 \wedge n > 0$
 $\Rightarrow \langle 1=1 \wedge 13=13 \text{ is vacuously true} \rangle$
 $1 = 1 \wedge 13 = 13 \wedge i = i_0 \wedge n > 0$
 $\Rightarrow \langle A \wedge B \wedge C \wedge D \Rightarrow A \wedge B \wedge C \rangle$
 $1 = 1 \wedge 13 = 13 \wedge n > 0$
 $\Rightarrow \langle 1 = 1 \Rightarrow (i = 1)^{[1/i]}, 13 = 13 \Rightarrow (x = 13)^{[13/x]} \rangle$
 $(i = 1 \wedge x = 13 \wedge n > 0)^{[1/i]}[^{13/x}]$
 $\Rightarrow \langle \text{clearly} \rangle$
 RHS

2.2 Implication 2: $(6) \sqsubseteq r, x : [Inv^{13/x}, Inv]$

To prove w-pre we need to prove: $pre \Rightarrow pre'$

To prove: $i = 1 \wedge n > 0 \wedge x = 13 \Rightarrow Inv^{13/r}$

Proof:

$$LHS = i = 1 \wedge n > 0 \wedge x = 13$$

$$\Rightarrow \langle A \wedge B \wedge C \Rightarrow A \wedge B \rangle$$

$$i = 1 \wedge x = 13$$

$$\Rightarrow \langle \text{We know that 13 is the 1st emirp from our definition of emirp, also } 13 \geq r \text{ in this case} \rangle$$

$$i = 1 \wedge Emirp(13, 1) \wedge x = 13 \wedge 13 \geq r$$

$$\Rightarrow \langle \text{This is our Inv with 13 substituted for x} \rangle$$

$$Inv^{13/x}$$

$$\Rightarrow \langle \text{clearly} \rangle$$

$$RHS$$

2.3 Implication 3: $r, x : [Inv^{13/r}, Inv] \sqsubseteq r := 13$

To prove: $r = r_0 \wedge Inv^{13/r} \Rightarrow Inv^{13/r}$

Proof:

$$LHS = r = r_0 \wedge Inv^{13/r}$$

$$\Rightarrow \langle A \wedge B \Rightarrow A \rangle$$

$$Inv^{13/r}$$

$$\Rightarrow \langle \text{clearly} \rangle$$

$$RHS$$

2.4 Implication 4: $Inv \wedge i = n \sqsubseteq Emirp(r, n)$

To prove: $Inv \wedge i = n \Rightarrow Emirp(r, n)$

Proof:

$$\begin{aligned} LHS &= Inv \wedge i = n \\ \Rightarrow \langle \text{Expanding the Invariant} \rangle \\ &Emirp(r, i) \wedge x \geq r \wedge i = n \\ \Rightarrow \langle \text{Combining conjuncts} \rangle \\ &Emirp(r, n) \wedge x \geq r \\ \Rightarrow \langle A \wedge B \Rightarrow A \rangle \\ &Emirp(r, n) \\ \Rightarrow \langle \text{Clearly} \rangle \\ &RHS \end{aligned}$$

2.5 Implication 5: $(10) \sqsubseteq skip$

To prove skip, we need to prove $pre \Rightarrow post^{[r_0/r]}$

To prove: $Emirp(r, n) \Rightarrow Emirp(r, n)^{[r_0/r]}$

Proof:

$$\begin{aligned} LHS &= Emirp(r, n) \\ \Rightarrow \langle \text{Since } r_0 \text{ is the value of } r \text{ in the precondition, } r = r_0 \text{ in the precondition} \rangle \\ &Emirp(r_0, n) \\ \Rightarrow \langle \text{clearly} \rangle \\ &RHS \end{aligned}$$

2.6 Implication 6: \sqsubseteq

To prove: $BLAH$

Proof:

$$LHS = BLAH$$

$$\Rightarrow \langle BLAH \rangle$$

$$BLAH$$

$$\Rightarrow \langle BLAH \rangle$$

$$BLAH$$

$$\Rightarrow \langle BLAH \rangle$$

$$BLAH$$

$$\Rightarrow \langle BLAH \rangle$$

$$RHS$$

2.7 Implication 7: $Inv^{[x+1/x]}, Inv^{[x+1/x]} \wedge a = 1 \sqsubseteq a := 1$

To prove: $a = a_0 \wedge Inv^{[x+1/x]} \Rightarrow (a = 1 \wedge Inv^{[x+1/x]})[1/a]$

Proof:

$$LHS = a = a_0 \wedge Inv^{[x+1/x]}$$

$$\Rightarrow \langle 1=1 \text{ is vacuously true} \rangle$$

$$1 = 1 \wedge a = a_0 \wedge Inv^{[x+1/x]}$$

$$\Rightarrow \langle A \wedge B \wedge C \Rightarrow A \wedge B \rangle$$

$$1 = 1 \wedge Inv^{[x+1/x]}$$

$$\Rightarrow \langle 1 = 1 \Rightarrow (a = 1 \wedge Inv^{[x/x+1]})[1/i] \rangle \text{ (Since, Inv does not involve a)}$$

$$(a = 1 \wedge Inv^{[x+1/x]})[1/a]$$

$$\Rightarrow \langle \text{clearly} \rangle$$

$$RHS$$

2.8 Implication 8:

$$[Inv^{[x+1]/x}] \wedge a = 1, post(14)] \sqsubseteq [a = 1 \wedge x > 0, post(14)]$$

To prove: $Inv^{[x+1]/x} \wedge a = 1 \Rightarrow a = 1 \wedge x > 0$

Proof:

$$LHS = Inv^{[x+1]/x} \wedge a = 1$$

\Rightarrow \langle Expanding Inv and performing substitution \rangle

$$Emirp(r, n) \wedge x + 1 \geq r \wedge a = 1$$

\Rightarrow \langle Since x and r starts at 13 and we are incrementing x , $x > 0$ \rangle

$$x > 0 \wedge a = 1$$

\Rightarrow \langle Clearly \rangle

$$RHS$$

2.9 Implication 9: \sqsubseteq

To prove: $BLAH$

Proof:

$$LHS = BLAH$$

\Rightarrow \langle BLAH \rangle

$$BLAH$$

\Rightarrow \langle BLAH \rangle

$$BLAH$$

\Rightarrow \langle BLAH \rangle

$$BLAH$$

\Rightarrow \langle BLAH \rangle

$$RHS$$

2.10 Implication 10: $[Inv, Inv] \sqsubseteq skip$

To prove: $Inv \Rightarrow Inv^{[r_0/r]}$

Proof:

$LHS = Inv$

$\Rightarrow \langle \text{Since } r_0 \text{ is the value of } r \text{ in the pre-condition, thus in precondition, } r_0 = r \rangle$
 $Inv^{[r_0/r]}$

$\Rightarrow \langle \text{Clearly} \rangle$

RHS

2.11 Implication 11: $(18) \sqsubseteq s := 0$

To prove: $s = s_0 \wedge (a = 1 \wedge \neg \exists k \in 2..(x-1) (x \bmod k = 0)) \vee$
 $(a = 0 \wedge \exists k \in 2..(x-1) (x \bmod k = 0)) \Rightarrow s = 0^{[0/s]}$

Proof:

$LHS = s = s_0 \wedge (a = 1 \wedge \neg \exists k \in 2..(x-1) (x \bmod k = 0)) \vee$
 $(a = 0 \wedge \exists k \in 2..(x-1) (x \bmod k = 0))$

$\Rightarrow \langle 0=0 \text{ is vacuously true} \rangle$

$0 = 0 \wedge s = s_0 \wedge (a = 1 \wedge \neg \exists k \in 2..(x-1) (x \bmod k = 0)) \vee$
 $(a = 0 \wedge \exists k \in 2..(x-1) (x \bmod k = 0))$

$\Rightarrow \langle A \wedge B \wedge C \Rightarrow A \rangle$

$0 = 0$

$\Rightarrow \langle 0 = 0 \Rightarrow (s = 0)^{[1/s]} \rangle$

$(s = 0)^{[0/s]}$

$\Rightarrow \langle \text{clearly} \rangle$

RHS

2.12 Implication 12: $[pre(19), \text{reversen function post condition}] \sqsubseteq$ $[\text{reversen function pre condition}, \text{reversen function post condition}]$

2.13 Implication 13: $[pre(21), pre(21) \wedge b = 1] \sqsubseteq b := 1$

To prove: $b = b_0 \wedge pre(21) \Rightarrow (pre(21) \wedge b = 1)[^1/b]$

Proof:

$$\begin{aligned}
 LHS &= b = b_0 \wedge pre(21) \\
 \Rightarrow &\langle 1=1 \text{ is vacuously true} \rangle \\
 &1 = 1 \wedge b = b_0 \wedge pre(21) \\
 \Rightarrow &\langle A \wedge B \wedge C \Rightarrow A \wedge B \rangle \\
 &1 = 1 \wedge pre(21) \\
 \Rightarrow &\langle 1 = 1 \Rightarrow (b = 1 \wedge pre(21))[^1/b] \text{ since } b \text{ does not appear in } pre(21) \rangle \\
 &(b = 1 \wedge pre(21))[^1/b] \\
 \Rightarrow &\langle \text{Clearly} \rangle \\
 RHS &
 \end{aligned}$$

2.14 Implication 14:

$$\begin{aligned}
 &[\text{reversen function post condition} \wedge b = 1, post(24)] \sqsubseteq \\
 &[s > 0 \wedge b = 1, post(24)]
 \end{aligned}$$

To prove: $\text{reversen function post condition} \wedge b = 1 \Rightarrow s > 0 \wedge b = 1$

Proof:

$$\begin{aligned}
 LHS &= \text{reversen function post condition} \wedge b = 1 \\
 \Rightarrow &\langle s = \text{reversen function post condition} \Rightarrow s \neq 0 \rangle \\
 &s > 0 \wedge b = 1 \\
 \Rightarrow &\langle \text{clearly} \rangle \\
 RHS &
 \end{aligned}$$

2.15 Implication 15: \sqsubseteq

To prove: $BLAH$

Proof:

$$\begin{aligned} LHS &= BLAH \\ \Rightarrow \langle BLAH \rangle \\ BLAH \\ \Rightarrow \langle BLAH \rangle \\ BLAH \\ \Rightarrow \langle BLAH \rangle \\ BLAH \\ \Rightarrow \langle BLAH \rangle \\ RHS \end{aligned}$$

2.16 Implication 16: $[Inv^{i+1}/i][x/r], Inv \sqsubseteq i := i + 1; r := x$

To prove: $i = i_0 \wedge r = r_0 \wedge Inv^{i+1}/i[x/r] \Rightarrow Inv^{i+1}/i[x/r]$

Proof:

$$\begin{aligned} LHS &= i = i_0 \wedge r = r_0 \wedge Inv^{i+1}/i[x/r] \\ \Rightarrow \langle A \wedge B \wedge C \Rightarrow A \rangle \\ Inv^{i+1}/i[x/r] \\ \Rightarrow \langle \text{Cleary} \rangle \\ RHS \end{aligned}$$

2.17 Implication 17: \sqsubseteq

To prove: $BLAH$

Proof:

$LHS = BLAH$

$\Rightarrow \langle BLAH \rangle$

$BLAH$

$\Rightarrow \langle BLAH \rangle$

$BLAH$

$\Rightarrow \langle BLAH \rangle$

$BLAH$

$\Rightarrow \langle BLAH \rangle$

RHS

2.18 Implication 18: $[Inv, Inv] \sqsubseteq skip$

To prove: $Inv \Rightarrow Inv^{[r_0/r]}$

Proof:

$LHS = Inv$

$\Rightarrow \langle \text{Since } r_0 \text{ is the value of } r \text{ in the pre-condition, thus in precondition, } r_0 = r \rangle$

$Inv^{[r_0/r]}$

$\Rightarrow \langle \text{Clearly} \rangle$

RHS

2.19 Implication 19: $[a = 1 \wedge r > 0, a = 1 \wedge r > 0 \wedge j = 2] \sqsubseteq j := 2$

To prove: $j = j_0 \wedge a = 1 \wedge r > 0 \Rightarrow (a = 1 \wedge r > 0 \wedge j = 2)^{[2/j]}$

Proof:

$$\begin{aligned}
 & LHS = j = j_0 \wedge a = 1 \wedge r > 0 \\
 \Rightarrow & \langle 2=2 \text{ is vacuously true} \rangle \\
 & j = j_0 \wedge a = 1 \wedge r > 0 \wedge 2 = 2 \\
 \Rightarrow & \langle 2=2 \Rightarrow (j = 2 \wedge a = 1 \wedge r > 0)^{[2/j]} \rangle \\
 & (j = 2 \wedge a = 1 \wedge r > 0)^{[2/j]} \\
 \Rightarrow & \langle \text{Clearly} \rangle \\
 & RHS
 \end{aligned}$$

2.20 Implication 20: \sqsubseteq

To prove: $BLAH$

Proof:

$$\begin{aligned}
 & LHS = BLAH \\
 \Rightarrow & \langle BLAH \rangle \\
 & BLAH \\
 \Rightarrow & \langle BLAH \rangle \\
 & BLAH \\
 \Rightarrow & \langle BLAH \rangle \\
 & BLAH \\
 \Rightarrow & \langle BLAH \rangle \\
 & RHS
 \end{aligned}$$

2.21 Implication 21: $[Inv_2, Inv_2] \sqsubseteq skip$

To prove: $Inv_2 \Rightarrow Inv_2[r_0/r]$

Proof:

$$LHS = Inv_2$$

$$\Rightarrow \langle \text{Since } r_0 \text{ is the value of } r \text{ in the pre-condition, thus in precondition, } r_0 = r \rangle$$

$$Inv_2[r_0/r]$$

$$\Rightarrow \langle \text{Clearly} \rangle$$

$$RHS$$

2.22 Implication 22: $[Inv_2 \wedge j = r, post(13)] \sqsubseteq [post(13), post(13)]$

To prove: $Inv_2 \wedge j = r \Rightarrow post(3)$

Proof:

$$LHS = (a = 1 \wedge \neg \exists k \in 2..(j-1) (j \bmod k = 0))$$

$$\vee (a = 0 \wedge \exists k \in 2..(j-1) (j \bmod k = 0) \wedge j = r)$$

$$\Rightarrow \langle \text{Combining conjuncts} \rangle$$

$$(a = 1 \wedge \neg \exists k \in 2..(r-1) (r \bmod k = 0))$$

$$\vee (a = 0 \wedge \exists k \in 2..(r-1) (r \bmod k = 0))$$

$$\Rightarrow \langle \text{Clearly} \rangle$$

$$RHS$$

2.23 Implication 23: $[post(3), post(3)] \sqsubseteq skip$

To prove: $post(3) \Rightarrow post(3)[r_0/r]$

Proof:

$$LHS = post(3)$$

$$\Rightarrow \langle \text{Since } r_0 \text{ is the value of } r \text{ in the pre-condition, thus in precondition, } r_0 = r \rangle$$

$$post(3)[r_0/r]$$

$$\Rightarrow \langle \text{Clearly} \rangle$$

$$RHS$$

2.24 Implication 24: $[Inv_2^{[j+1]/j}, Inv_2] \sqsubseteq j := j + 1$

To prove: $j = j_0 \wedge Inv_2^{[j+1]/j} \Rightarrow Inv_2^{[j+1]/j}$

Proof:

$$\begin{aligned}
 LHS &= j = j_0 \wedge Inv_2^{[j+1]/j} \\
 \Rightarrow &\langle A \wedge B \Rightarrow A \rangle \\
 &Inv_2^{[j+1]/j} \\
 \Rightarrow &\langle \text{Clearly} \rangle \\
 RHS &
 \end{aligned}$$

2.25 Implication 25: \sqsubseteq

To prove: $BLAH$

Proof:

$$\begin{aligned}
 LHS &= BLAH \\
 \Rightarrow &\langle BLAH \rangle \\
 &BLAH \\
 \Rightarrow &\langle BLAH \rangle \\
 &BLAH \\
 \Rightarrow &\langle BLAH \rangle \\
 &BLAH \\
 \Rightarrow &\langle BLAH \rangle \\
 RHS &
 \end{aligned}$$

2.26 Implication 26: $[Inv_2^{[j+1]/j}][^0/a] \sqsubseteq post(11)$

To prove: $a = a_0 \wedge Inv_2^{[j+1]/j}[^0/a] \Rightarrow post(11)$

Proof:

$$\begin{aligned}
 LHS &= a = a_0 \wedge Inv_2^{[j+1]/j}[^0/a] \\
 &\Rightarrow \langle \text{Removing conjuncts} \rangle \\
 &\quad Inv_2^{[j+1]/j}[^0/a] \\
 &\Rightarrow \langle \text{Clearly} \rangle \\
 &\quad RHS
 \end{aligned}$$

2.27 Implication 27: $(12) \sqsubseteq [Inv_2, post(11)]$

To prove: $BLAH$

Proof:

$$\begin{aligned}
 LHS &= BLAH \\
 &\Rightarrow \langle BLAH \rangle \\
 &\quad BLAH \\
 &\Rightarrow \langle BLAH \rangle \\
 &\quad BLAH \\
 &\Rightarrow \langle BLAH \rangle \\
 &\quad BLAH \\
 &\Rightarrow \langle BLAH \rangle \\
 &\quad RHS
 \end{aligned}$$

2.28 Implication 28: $[Inv_2, post(11)] \sqsubseteq skip$

To prove: $Inv_2[j^{+1}/j] \Rightarrow post(11)[r_0/r]$

Proof:

$LHS = Inv_2[j^{+1}/j]$
 $\Rightarrow \langle \text{Since } r_0 \text{ is the value of } r \text{ in the pre-condition, thus in precondition, } r_0 = r \rangle$
 $Inv_2[j^{+1}/j][r_0/r]$
 $\Rightarrow \langle post(11) = Inv_2[j^{+1}/j] \rangle$
 RHS

3 Task 3 - C Code

```
1 #include <stdio.h>
2 #include "reverse.h"
3 #define USEGMP
4 unsigned long emirp(unsigned long n);
5 void isPrime(unsigned long r, int *a);
6
7 int main (int argc, char* argv[]){
8     unsigned long n;
9     if(scanf("%lu", &n)==1)
10         printf("%lu\n",emirp(n));
11 }
12
13 unsigned long emirp(unsigned long n) {
14     int i = 1;
15     unsigned long x = 13;
16     unsigned long r = 13;
17     while (i != n) {
18         x = x + 1;
19         int a = 1;
20         isPrime(x, &a);
21         if (a == 1) {
22             unsigned long s = 0;
23             reversen(x, &s);
24             int b = 1;
25             isPrime(s, &b);
26             if (b == 1 && s != x)
27                 i = i + 1;
```

```

28             r = x;
29         }
30     }
31     return r;
32 }
33
34 void isPrime(unsigned long r, int *a) {
35     unsigned long j = 2;
36     while (j != r) {
37         if (r % j == 0)
38             *a = 0;
39         j = j + 1;
40     }
41 }

```

- Write something about how the C code relates.
- Compare with examples