Assignment 2 - Emirps

Rahil Agrawal z5165505
Aditya Karia zXXXXXXX

COMP2111 18s1

1 Task 1 - Specification Statement

Notes:
-Write neatly
-make sure grammar is correct
-look at examples for default spec structure.

Define an Emirp using 2 functions - reverse and prime. Make these functions match with their given specs in order to help prove implications.

Pre condition: n is a positive number - $n > 0$

Post condition $EMIRP(r, n)$ where r is the n^{th} emirp(where emirp is as defined above). Therefore our program can be specified by:
proc EMIRP(**value** n, **result** r) $\cdot$
$\llcorner n, r : [\ n > 0, Emirp(r, n)\ ]_{\lrcorner(1)}$

2 Task 2 - Derivation

proc EMIRP(**value** n, **result** r) $\cdot$

$\llcorner n, r : [\ n > 0, Emirp(r, n)\ ]\lrcorner_{(1)}$

(1) $\sqsubseteq$ ⟨**c-frame**⟩

$\llcorner r : [\ n > 0, Emirp(r, n)\ ]\lrcorner_{(2)}$

(2) $\sqsubseteq$ ⟨**i-loc**⟩

$\llcorner i, r : [\ n > 0, Emirp(r, n)\ ]\lrcorner_{(3)}$

(3) $\sqsubseteq$ ⟨**seq**⟩

$\llcorner i, r : [\ n > 0, i = 1 \wedge n > 0\ ]\lrcorner_{(4)};$

$\llcorner i : [\ i = 1 \wedge n > 0, Emirp(r, n)\ ]\lrcorner_{(5)}$

(4) $\sqsubseteq$ ⟨**c-frame**⟩

$i : [\ n > 0, i = 1 \wedge n > 0\ ]$

$\sqsubseteq$ ⟨**ass - (1)**⟩

$i := 1$

(5) $\sqsubseteq$ ⟨**seq**⟩

$\llcorner i, r : [\ i = 1 \wedge n > 0, Inv\ ]\lrcorner_{(6)};$

$\llcorner i, r : [\ Inv, Inv \wedge i = n\ ]\lrcorner_{(7)};$

$\llcorner i, r : [\ Inv \wedge i = n, Emirp(r, n)\ ]\lrcorner_{(8)}$

(6) $\sqsubseteq$ ⟨**w-pre, c-frame - (2)**⟩

$\llcorner r : [\ Inv[^{13}/_r], Inv\ ]\lrcorner_{(9)}$

$\sqsubseteq$ ⟨**ass - (3)**⟩

$r := 13$

(7) $\sqsubseteq$ ⟨**while**⟩

while $i \neq n$ **do**

$\llcorner i, r : [\ Inv \wedge i \neq n, Inv\ ]\lrcorner_{(10)}$

od;

(8) $\sqsubseteq$ ⟨**w-pre**⟩

$\llcorner r : [\ \text{EMIRP}(r, n), \text{EMIRP}(r, n)\ ]\lrcorner_{(11)}$

$\sqsubseteq$ ⟨**skip - (4)**⟩

skip

(10) $\sqsubseteq$ ⟨**seq**⟩

$\llcorner r : [\ Inv \wedge i \neq n, Inv[^{r+1}/_r]\ ]\lrcorner_{(11)};$

$\llcorner r : [\ Inv[^{r+1}/_r], Inv\ ]\lrcorner_{(12)}$

$(11) \sqsubseteq \quad \langle \textbf{ass - (5)} \rangle$

$r := r + 1$

$(12) \sqsubseteq \quad \langle \textbf{seq} \rangle$

$\llcorner a, i, r : \left[\ Inv[^{r+1}/_r], Inv[^{r+1}/_r] \wedge a = 1\ \right]\lrcorner_{(14)};$

$\llcorner a, i, r : \left[\ Inv[^{r+1}/_r] \wedge a = 1, Inv\ \right]\lrcorner_{(15)}$

$(14) \sqsubseteq \quad \langle \textbf{c-frame} \rangle$

$\llcorner a : \left[\ Inv[^{r+1}/_r], Inv[^{r+1}/_r] \wedge a = 1\ \right]\lrcorner_{(16)}$

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

$a := 1$

$(15) \sqsubseteq \quad \langle \textbf{seq} \rangle$

$$\llcorner a, i, r : \left[\begin{array}{l} Inv[^{r+1}/_r] \wedge a = 1, (a = 1 \wedge \nexists k \in 2..(r-1)\,(r \textbf{ mod } k = 0)) \\ \vee (a = 0 \wedge \exists k \in 2..(r-1)\,(r \textbf{ mod } k = 0))) \end{array} \right]\lrcorner_{(17)};$$

$$\llcorner a, i, r : \left[\begin{array}{l} (a = 1 \wedge \nexists k \in 2..(r-1)\,(r \textbf{ mod } k = 0)) \\ \vee (a = 0 \wedge \exists k \in 2..(r-1)\,(r \textbf{ mod } k = 0))), Inv \end{array} \right]\lrcorner_{(18)}$$

$(18) \sqsubseteq \quad \langle \textbf{if} \rangle$

if $a = 1$

then $\llcorner a, i, r : [a = 1 \wedge pre(18), post(18)]\lrcorner_{(19)}$

else $\llcorner p : [a \neq 1 \wedge pre(18), post(18)]\lrcorner_{(20)}$

fi

$(19) \sqsubseteq \quad \langle \textbf{i-loc} \rangle$

$a, i, r, s : \left[\ pre(19), post(19)\ \right]$

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

$\llcorner s : \left[\ pre(19), post(19) \wedge s = 0\ \right]\lrcorner_{(21)};$

$\llcorner a, i, r, s : \left[\ post(19) \wedge s = 0, post(19)\ \right]\lrcorner_{(22)}$

$(21) \sqsubseteq \quad \langle \textbf{ass - (7)} \rangle$

$s := 0$

$(22) \sqsubseteq \quad \langle \textbf{seq} \rangle$

$\llcorner s : \left[\ pre(22), \text{reversen function post condition}\ \right]\lrcorner_{(23)};$

$\llcorner a, i, r, s : \left[\ \text{reversen function post condition}, post(22)\ \right]\lrcorner_{(24)}$

$(23) \sqsubseteq \quad \langle \textbf{w-pre} \rangle$

$s : \left[\ \text{reverse function pre condition}, \text{reversen function post condition}\ \right]$

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

$\llcorner s : \left[\ pre(19), post(19) \wedge s = 0\ \right]\lrcorner_{(21)}$

We gather the code for the procedure body of EMIRP:

3 Task 3 - C Code

```
#include <stdio.h>
#include ”reverse.h”

unsigned long emirp(unsigned long n);
void isPrime(unsigned long r, int *a);

int main (int argc, char* argv[]){
        unsigned long n;
        if(scanf(”%lu”, &n)==1)
            printf(”%lu\n”,emirp(n));
}

/*
var i := 1
r := 13
while i != n do
    r := r + 1
    var a := 1
    isPrime(r,a)
    if a = 1 then
        var s := 0
        reversen(r,s)
        var b := 1
        isPrime(s, b)
        if b = 1 && s != r then
            i = i + 1
od

*/
unsigned long emirp(unsigned long n) {
        int i = 1;
        unsigned long r = 13;
    while (i != n) {
        r = r + 1;
        int a = 1;
        isPrime(r, &a);
        if (a == 1) {
            unsigned long s = 0;
            reversen(r, &s);
```

```
            int b = 1;
            isPrime(s, &b);
            if (b == 1 && s != r) {
                i = i + 1;
            }
        }
    }
        return r;
}

/*
var j := 0
while j != r do
    if r mod j = 0 then
        a = 0
    j := j + 1
od
*/
void isPrime(unsigned long r, int *a) {
    unsigned long j = 2;
    while (j != r) {
        if (r % j == 0) {
            *a = 0;
        }
        j = j + 1;
    }
}
```

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