Assignment 2 - Emirps

Rahil Agrawal z5165505
Aditya Karia zXXXXXXX

COMP2111 18s1

1 Task 1 - Specification Statement

The spec

2 Task 2 - Derivation

$$
\begin{array}{ll}
 & \textbf{proc } \textsc{emirp}(\textbf{value } n, \textbf{result } r) \cdot \\
 & \quad \llcorner n, r : [\ n > 0, Emirp(r, n)\ ] \lrcorner_{(1)} \\
(1) \sqsubseteq & \quad \langle \textbf{c-frame} \rangle \\
 & \llcorner r : [\ n > 0, Emirp(r, n)\ ] \lrcorner_{(2)} \\
(2) \sqsubseteq & \quad \langle \textbf{i-loc} \rangle \\
 & \llcorner i, r : [\ n > 0, Emirp(r, n)\ ] \lrcorner_{(3)} \\
(3) \sqsubseteq & \quad \langle \textbf{seq} \rangle \\
 & \llcorner i, r : [\ n > 0, i = 0 \wedge n > 0\ ] \lrcorner_{(4)}; \\
 & \llcorner i : [\ i = 0 \wedge n > 0, Emirp(r, n)\ ] \lrcorner_{(5)} \\
(4) \sqsubseteq & \quad \langle \textbf{c-frame} \rangle \\
 & \llcorner i : [\ n > 0, i = 1 \wedge n > 0\ ] \lrcorner \\
 \sqsubseteq & \quad \langle \textbf{ass - (1)} \rangle \\
 & i := 1 \\
(5) \sqsubseteq & \quad \langle \textbf{seq} \rangle \\
 & \llcorner i, r : [\ i = 1 \wedge n > 0, Inv\ ] \lrcorner_{(6)}; \\
 & \llcorner i, r : [\ Inv, Inv \wedge i = n\ ] \lrcorner_{(7)}; \\
 & \llcorner i, r : [\ Inv \wedge i = n, Emirp(r, n)\ ] \lrcorner_{(8)} \\
(6) \sqsubseteq & \quad \langle \textbf{w-pre, c-frame - (2)} \rangle \\
 & \llcorner r : [\ Inv[^{13}/_{r}], Inv\ ] \lrcorner_{(9)} \\
 \sqsubseteq & \quad \langle \textbf{ass - (3)} \rangle \\
 & r := 13 \\
(7) \sqsubseteq & \quad \langle \textbf{while} \rangle \\
 & \textbf{while } i \neq n \textbf{ do} \\
 & \quad \llcorner i, r : [\ Inv \wedge i \neq n, Inv\ ] \lrcorner_{(10)} \\
 & \textbf{od}; \\
(10) \sqsubseteq & \quad \langle \textbf{seq} \rangle \\
 & \llcorner r : [\ Inv \wedge i \neq n, Inv[^{r+1}/_{r}]\ ] \lrcorner_{(11)}; \\
 & \llcorner r : [\ Inv[^{r+1}/_{r}], Inv\ ] \lrcorner_{(12)}
\end{array}
$$

$$
\begin{array}{lll}
(11) \sqsubseteq & & \langle \textbf{i-loc} \rangle \\
& \llcorner a, i, r : \left[\; Inv[^{r+1}/_{r}], Inv \; \right] \lrcorner_{(13)} & \\
\sqsubseteq & & \langle \textbf{if} \rangle \\
& \textbf{if } l = r + 1 & \\
& \textbf{then } \llcorner p : [l = r + 1 \wedge pre(2), post(2)] \lrcorner_{(3)} & \\
& \textbf{else } \llcorner p : [l \neq r + 1 \wedge pre(2), post(2)] \lrcorner_{(4)} & \\
& \textbf{fi} &
\end{array}
$$

We gather the code for the procedure body of EMIRP:

3 Task 3 - C Code

```
1  #include <stdio.h>
2  #include "reverse.h"
3
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 /*
14 var i := 1
15 r := 13
16 while i != n do
17     r := r + 1
18     var a := 1
19     isPrime(r,a)
20     if a = 1 then
21         var s := 0
22         reversen(r,s)
23         var b := 1
24         isPrime(s, b)
25         if b = 1 && s != r then
26             i = i + 1
27 od
```

```
28
29 */
30 unsigned long emirp(unsigned long n) {
31         int i = 1;
32         unsigned long r = 13;
33     while (i != n) {
34         r = r + 1;
35         int a = 1;
36         isPrime(r, &a);
37         if (a == 1) {
38             unsigned long s = 0;
39             reversen(r, &s);
40             int b = 1;
41             isPrime(s, &b);
42             if (b == 1 && s != r) {
43                 i = i + 1;
44             }
45         }
46     }
47         return r;
48 }
49
50 /*
51 var j := 0
52 while j != r do
53     if r mod j = 0 then
54         a = 0
55     j := j + 1
56 od
57 */
58 void isPrime(unsigned long r, int *a) {
59     unsigned long j = 2;
60     while (j != r) {
61         if (r % j == 0) {
62             *a = 0;
63         }
64         j = j + 1;
65     }
66 }
```