

Assignment 2

z5019338

z5131048

August 15, 2018

1 Introduction

Given a positive integer number n , return the n 'th emirp. An emirp is a prime whose (base 10) reversal is also prime, but which is not a palindromic prime.

Firstly, let the set of all prime numbers be as *prime*, where *prime* denote as:

$$prime = \{p \in \mathbb{N}^+ \mid p > 1 \wedge \neg \exists k \in \mathbb{N}^+ . \{1, n\} : \frac{p}{k} \in \mathbb{N}^+\}$$

Let the sequence of all the prime numbers be as P_n , where P_n is denoted as

$$P_n = \langle 2, 3, 5, 7, 11, \dots \rangle_{n \in \mathbb{N}^+}$$

Given proc *reversen* as defined in the specification, it is going to be denoted as the mathematical function $res : \mathbb{N} \rightarrow \mathbb{N}$ as following:

$$res(n) = \sum_{i=0}^{\lfloor \log n \rfloor} S_i 10^i$$

where $n = \sum_{i=0}^{\lfloor \log n \rfloor} S_i 10^{\lfloor \log n \rfloor - i}$

2 The Derivation

Before deriving **proc** EMIRP(**value** $n : \mathbb{N}^+$, **result** $r : \mathbb{N}^+$), it needs to develop 2 procedure that help with proc EMIRP

proc ISPRIME(**value** $n : \mathbb{N}^+$, **result** $b : \mathbb{B}$)

proc NEXTPRIME(**value** $p : \mathbb{N}^+$, **result** $q : \mathbb{N}^+$).

proc NEXTPRIME(**value** p , **result** q) ·

$\sqcup p, q : [p \in P_i, i := i_0 + 1 \wedge q := p_i] \neg(A)$

$$\begin{aligned}
(A) &\sqsubseteq \langle \mathbf{i-con} \rangle \\
&\quad \sqsubseteq \text{var } con \ b \cdot p, q : [p \in P_i \wedge i = b, q := p_i \wedge i := i_0 + 1] \dashv(B) \\
(B) &\sqsubseteq \langle \mathbf{s-post} \rangle \\
&\quad p, q : [p \in P_i \wedge i = b, q := P_i \wedge i = b + 1] \\
&\sqsubseteq \langle \mathbf{seq2} \rangle \\
&\quad i := i + 1; q := P_i
\end{aligned}$$

$$\begin{aligned}
&\mathbf{proc} \text{ ISPRIME}(\mathbf{value} \ n, \mathbf{result} \ r) \cdot \\
&\quad \sqsubseteq n, r : [n > 0, b \Leftrightarrow n \in \text{prime}] \dashv(I) \\
(I) &\sqsubseteq \langle \mathbf{i-loc} \rangle \\
&\quad \sqsubseteq \text{var } k \cdot n, b, k : [n > 0, b \Leftrightarrow n \in \text{prime}] \dashv(II) \\
(II) &\sqsubseteq \langle \mathbf{c-frame, seq, split into initialisation and loop} \rangle \\
&\quad \sqsubseteq b : [n > 0, Inv] \dashv(III) \\
&\quad \sqsubseteq n, b, k : [Inv, b \Leftrightarrow n \in \text{prime}] \dashv(IV)
\end{aligned}$$

Where the loop invariant is defined as:

$$Inv = \left(\begin{array}{l} (n = 1 \Rightarrow b = \text{False}) \vee \\ 2 < k \leq n \wedge (\neg \exists k \in 2..(n-1) (p \mid k) \Leftrightarrow n \in \text{prime} \wedge b = \text{True}) \\ \vee (\exists k \in 2..(n-1) (p \mid k) \Leftrightarrow n \notin \text{prime} \wedge b = \text{False}) \end{array} \right)$$

$$\begin{aligned}
(III) &\sqsubseteq \langle \mathbf{seq, ass} \rangle \\
&\quad \sqsubseteq b, k : [n > 0, Inv^{[2, \text{True}]/k, b}] \dashv_{\spadesuit} \\
&\quad \mathbf{if} \ n = 1 \ \mathbf{then} \\
&\quad \quad \sqsubseteq n, b, k : [n > 0 \wedge n = 1, Inv^{[1, \text{False}]/n, b}] \dashv_{\heartsuit} \\
&\quad \mathbf{else;} \\
&\quad \quad \sqsubseteq n, b, k : [n > 0 \wedge n \neq 1, Inv^{[1, \text{False}]/n, b}] \dashv_{\diamond} \\
&\quad \mathbf{fi} \\
&\quad \spadesuit \sqsubseteq \langle \mathbf{ass, justified below in Sect. 2.1} \rangle \\
&\quad \quad b := \text{True}; \\
&\quad \quad k := 2; \\
&\quad \heartsuit \sqsubseteq \langle \mathbf{ass, justified below in Sect. 2.2} \rangle \\
&\quad \quad b := \text{False} \\
&\quad \diamond \sqsubseteq \langle \mathbf{skip} \rangle \\
&\quad \quad \text{skip}
\end{aligned}$$

$$\begin{aligned}
(IV) &\sqsubseteq \langle \mathbf{s\text{-}post}, \text{justified below in Sect. 2.3} \rangle \\
&\quad n, b, k : [\text{Inv}, \text{Inv} \wedge k = n] \\
&\sqsubseteq \langle \mathbf{c\text{-}frame}, \mathbf{while} \rangle \\
&\quad \mathbf{while } k < n \mathbf{ do} \\
&\quad \quad \sqsubset n, b, k : [\text{Inv} \wedge k < n, \text{Inv}^{[k+1]/k}] \neg(V) \\
&\quad \mathbf{od;} \\
(V) &\sqsubseteq \langle \mathbf{seq}, \mathbf{c\text{-}frame} \rangle \\
&\quad \sqsubset n, b, k : [\text{Inv} \wedge k < n, \text{Inv}^{[k+1]/k}] \neg(VI) \\
&\quad \sqsubset k : [\text{Inv} \wedge k < n, \text{Inv}^{[k+1]/k}] \neg(VII) \\
(VI) &\sqsubseteq \langle \mathbf{if} \rangle \\
&\quad \mathbf{if } n|k \mathbf{ then} \\
&\quad \quad \sqsubset n, b, k : [n|k \wedge \text{Inv} \wedge k < n, \text{Inv}^{[k+1]/k}] \neg(VIII) \\
&\quad \mathbf{else;} \\
&\quad \quad \sqsubset n, k : [n \nmid k \wedge \text{Inv} \wedge k < n, \text{Inv}^{[k+1]/k}] \neg(IX) \\
&\quad \mathbf{fi} \\
(VIII) &\sqsubseteq \langle \mathbf{ass} \rangle \\
&\quad b := \text{False}; \\
(IX) &\sqsubseteq \langle \mathbf{skip} \rangle \\
&\quad \text{skip}; \\
(VII) &\sqsubseteq \langle \mathbf{ass} \rangle \\
&\quad k := k + 1;
\end{aligned}$$

2.1 Proof of $\spadesuit \sqsubseteq b := \text{True}; k := 2;$

We need to prove validity

$$n > 0 \Rightarrow \text{Inv}^{[2, \text{True}/k, b]}$$

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

$$\begin{aligned}
&n > 0 \Rightarrow \\
&\left(\begin{aligned}
&(n = 1 \Rightarrow b = \text{False}) \vee \\
&2 < 2 \leq n \wedge (\neg \exists 2 \in 2..(n-1) (p | k) \Leftrightarrow n \in \text{prime} \wedge \text{True} = \text{True}) \\
&\vee (\exists k \in 2..(n-1) (p | k) \Leftrightarrow n \notin \text{prime} \wedge b = \text{False})
\end{aligned} \right)
\end{aligned}$$

Clearly, we have established validity of the **second conjunct** of the RHS. It's obviously true. As long as any disjunct is true, the rest is true

2.2 Proof of $\heartsuit \sqsubseteq b := \text{False};$

We need to prove validity

$$n > 0 \wedge n = 1 \Rightarrow \text{Inv}^{[1, \text{False}] / n, b}$$

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

$$n > 0 \wedge n = 1 \Rightarrow \left(\begin{array}{l} (n = 1 \Rightarrow \text{False} = \text{False}) \vee \\ 2 < k \leq n \wedge (\neg \exists k \in 2..(n-1) (p \mid k) \Leftrightarrow n \in \text{prime} \wedge b = \text{True}) \\ \vee (\exists k \in 2..(n-1) (p \mid k) \Leftrightarrow n \notin \text{prime} \wedge b = \text{False}) \end{array} \right)$$

Clearly, we have established validity of the **first conjunct** of the RHS. It's obviously true. As long as any disjunct is true, the rest is true

2.3 Proof of $(IV) \sqsubseteq \text{Inv} \Rightarrow \text{Inv} \wedge k = n$

Expanding the definitions and performing the substitution yields

$$\left(\begin{array}{l} (n = 1 \Rightarrow b = \text{False}) \vee \\ 2 < k \leq n \wedge (\neg \exists k \in 2..(n-1) (p \mid k) \Leftrightarrow n \in \text{prime} \wedge b = \text{True}) \\ \vee (\exists k \in 2..(n-1) (p \mid k) \Leftrightarrow n \notin \text{prime} \wedge b = \text{False}) \end{array} \right) \Rightarrow \left(\begin{array}{l} (n = 1 \Rightarrow b = \text{False}) \vee \\ 2 < k \leq n \wedge (\neg \exists k \in 2..(n-1) (p \mid k) \Leftrightarrow n \in \text{prime} \wedge b = \text{True}) \\ \vee (\exists k \in 2..(n-1) (p \mid k) \Leftrightarrow n \notin \text{prime} \wedge b = \text{False}) \wedge k = n \end{array} \right)$$

According there is 3 disjunction of the LHS, there are 3 cases to consider:

1. $n > 0 \wedge n = 1 \Rightarrow \text{Inv}^{[1, 2, \text{False}] / n, 2, b}$: as prove above, the fist disjunct of the RHS follows immediately.
2. $2 \leq k \leq n-1 \wedge p \nmid k \Rightarrow \text{Inv}^{[k+1, \text{True}] / k, b}$: we prove the second disjunct of the RHS. As $2 \leq k \leq n-1 \wedge p \nmid k \Rightarrow \neg \exists k \in 2..(n-1) (p \mid k) \Leftrightarrow n \in \text{prime} \wedge b = \text{True}$
3. $2 \leq k \leq n-1 \wedge p \mid k \Rightarrow \text{Inv}^{[k+1, \text{False}] / k}$: similarly, $2 \leq k \leq n-1 \wedge p \mid k \Rightarrow \exists k \in 2..(n-1) (p \mid k) \Leftrightarrow n \in \text{prime} \wedge b = \text{False}$

Clearly, we have established validity of the RHS. It's obviously true.

We gather the code for the procedure body of **ISPRIME**:

```

b := True;
if n = 1 then
  b := False;
else

```

```

        skip
    fi
    k := 2
    while k < n do
        if n = 1 then
            b := False;
        else
            skip
        fi
    od;

```

```

proc EMIRP(value n, result r) ·
     $\sqsubseteq n, r : \left[ \begin{array}{l} n > 0, \\ n = 0 \wedge r \in \text{prime} \wedge \text{rev}(r) \in \text{prime} \wedge r \neq \text{rev}(r) \end{array} \right] \neg(1)$ 
(1)  $\sqsubseteq$      $\langle \text{i-loc} \rangle$ 
     $\sqsubseteq \text{var } s \cdot n, r, s : \left[ \begin{array}{l} n > 0, \\ n = 0 \wedge r \in \text{prime} \wedge \text{rev}(r) \in \text{prime} \wedge r \neq \text{rev}(r) \end{array} \right] \neg(2)$ 
(2)  $\sqsubseteq$      $\langle \text{seq, split into initialisation and loop} \rangle$ 
     $\sqsubseteq s : [ n > 0, \text{Inv} ] \neg(3)$ 
     $\sqsubseteq n, r, s : [ \text{Inv}, n = 0 \wedge r \in \text{prime} \wedge \text{rev}(r) \in \text{prime} \wedge r \neq \text{rev}(r) ] \neg(4)$ 

```

Where the loop invariant is defined as:

$$\text{Inv} = (0 \leq i \leq n \wedge s = P_i \wedge \text{rev}(s) \in \text{prime} \wedge s \neq \text{rev}(s) \wedge r = P_{(n-i)})$$

where P is the prime numbers sequence.

```

(3)  $\sqsubseteq$      $\langle \text{ass} \rangle$ 
     $s : [ n > 0, \text{Inv}^{[13,n]/s}, n ]$ 
 $\sqsubseteq$      $\langle \text{ass, justified below in Sect. 2.4} \rangle$ 
    s := 13
(4)  $\sqsubseteq$      $\langle \text{s-post, justified below in Sect. 2.6} \rangle$ 
    n, r, s : [  $\text{Inv}^{[n-1]/n}, \text{Inv} \wedge n = 0$  ]
 $\sqsubseteq$      $\langle \text{while, seq, c-frame} \rangle$ 
    while n ≠ 0 do
         $\sqsubseteq \text{var } k, b \cdot n, r, s : [ \text{Inv} \wedge n \neq 0, \text{Inv}^{[n-1]/n} ] \neg(A)$ 
    od;

```

$$\begin{aligned}
& \sqsubseteq r : [\text{Inv} \wedge n = 0, \text{Inv}^{[0/n]}] \sqsubseteq_{(B)} \\
A & \sqsubseteq \langle \text{c-frame, seq} \rangle \\
& \sqsubseteq k, b : [\text{Inv} \wedge n \neq 0, \text{Inv}^{[n-1/n]}] \sqsubseteq_{(A1)} \\
& \sqsubseteq n, k, b, s : [\text{Inv} \wedge n \neq 0, \text{Inv}^{[n-1/n]}] \sqsubseteq_{(A2)} \\
& \sqsubseteq s : [\text{Inv} \wedge n \neq 0, \text{Inv}^{[n-1/n]}] \sqsubseteq_{(A3)} \\
A1 & \sqsubseteq \langle \dots, \text{proc, seq} \rangle \\
& \text{REVERSE}(s, k); \text{ISPRIME}(k, b); \\
A2 & \sqsubseteq \langle \text{if, where } g = (n \neq 0 \wedge k = \text{rev}(s) \wedge b \wedge k \neq s \vee n = 0) \rangle \\
& \text{if } g \text{ then} \\
& \quad \sqsubseteq n, k, s : [g \wedge \text{Inv} \wedge n \neq 0, \text{Inv}^{[n-1/n]}] \sqsubseteq_{(i)} \\
& \quad \text{else} \\
& \quad \quad \sqsubseteq n, k, s : [\neg g \wedge \text{Inv} \wedge n \neq 0, \text{Inv}^{[n-1/n]}] \sqsubseteq_{(ii)} \\
& \quad \text{fi} \\
i & \sqsubseteq \langle \text{ass} \rangle \\
& n := n - 1; \\
ii & \sqsubseteq \langle \text{skip} \rangle \\
& \text{skip} \\
A3 & \sqsubseteq \langle \dots, \text{proc} \rangle \\
& \text{NEXTPRIME}(s, s) \\
B & \sqsubseteq \langle \text{ass, justified below in Sect. 2.5} \rangle \\
& r := s;
\end{aligned}$$

2.4 Proof of $(3) \sqsubseteq s := 13$

We need to prove validity

$$n > 0 \Rightarrow \text{Inv}^{[13, n / s, n]}$$

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

$$\begin{aligned}
& n > 0 \Rightarrow \\
& (0 \leq n \leq n \wedge s = 13 \wedge \text{rev}(13) \in \text{prime} \wedge s \neq \text{rev}(13) \wedge r = P_{n-n})
\end{aligned}$$

Clearly, we have established the validity of those conjunct of the RHS with $s := 13$ and $n := n$. This whole conjuncts are obviously true when s initialises with 13.

2.5 Proof of $(B) \sqsubseteq r := s$

We need to prove validity

$$Inv \wedge n = 0 \Rightarrow Inv[{}^0/_n]$$

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

$$\begin{aligned} & (0 \leq i \leq n \wedge s = P_i \wedge rev(s) \in prime \wedge s \neq rev(s) \wedge r = P_{(n-i)} \wedge n = 0) \Rightarrow \\ & (0 \leq i \leq 0 \wedge s = P_0 \wedge rev(s) \in prime \wedge s \neq rev(s) \wedge r = P_n) \end{aligned}$$

Clearly, we should establish the validity of first, second and fifth conjunct of the RHS by replace n as 0 in the LHS. After substit n by 0 in the LHS, RHS is given with $n = 0$ implication.

2.6 Proof of $(4) \sqsubseteq Inv[{}^{n-1}/_n] \Rightarrow Inv \wedge n = 0$

Expanding the definitions and performing the substitution yields

$$\begin{aligned} & (0 \leq i \leq n - 1 \wedge s = P_i \wedge rev(s) \in prime \wedge s \neq rev(s) \wedge r = P_{(n-1-i)}) \Rightarrow \\ & (\textcolor{blue}{n} = 0 \wedge 0 \leq i \leq n \wedge s = P_i \wedge rev(s) \in prime \wedge s \neq rev(s) \wedge r = P_{(n-i)}) \end{aligned}$$

According the conjuncts of LHS, there are 2 cases to consider:

1. $n = 0 \wedge Inv[{}^0/_n]$: its validity as the previous proof.
2. $n \neq 0 \wedge Inv[{}^{n-1}/_n]$: while $n \neq 0$, the **first conjunct** can remove in the RHS and replace the rest n with n-1. This become identical with LHS, therefore, it is valid and true.

We gather the code for the procedure body of EMIRP:

```

     $s := 2;$ 
while  $n \neq 0$  do
     $reversen(s, k)$ 
    if  $k \in prime \wedge k \neq s$  then
         $n = n - 1$ 
    else
         $skip$ 
    fi
     $nextPrime(s, s)$ 
od;
```

3 The C Code

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <assert.h>
4  #include <gmp.h>
5  #define USEGMP
6
7  #include "reverse.h"
8  #define FALSE 0
9  #define TRUE 1
10
11 void emirp(mpz_t n, mpz_t r) {
12     /* 1. Initialize the beginning prime number s as 13 */
13     mpz_t s;
14     mpz_init(s);
15     mpz_set_ui(s, 13);
16
17     // decrement n till n equal to 0
18     while (mpz_cmp_ui(n, 0) != 0) {
19         /* 3. reverse s put into r */
20         reversen(s, r);
21         /* 4. check if r is prime number and r is not same as s */
22         if (mpz_probab_prime_p(r, 40) && mpz_cmp(r, s)) {
23             /* 5. decrement the n when r is prime */
24             mpz_sub_ui(n, n, 1);
25         }
26         /* 6. get the next prime number while n != 0 */
```



```

27     mpz_nextprime(s, s);
28 }
29 /* while n = 0, reverse r as nth prime is s and return r as result */
30 reversen(r, r);
31 }
32
33 int main() {
34     char inputStr[1024];
35     // mpz_t is the type defined for GMP integers.
36     // It is a pointer to the internals of the GMP integer data structure
37     mpz_t n;
38     int flag;
39
40     printf("Enter your number: ");
41     flag = scanf("%1023s", inputStr);
42     // NOTE: never ever write a call scanf ("%s", inputStr);
43     // You are leaving a security hole in your code.
44     assert(flag > 0);
45     // If flag is greater 0 then the operation /*
46
47     // 1. Initialize the number
48     mpz_init(n);
49     mpz_set_ui(n, 0);
50
51     // 2. Parse the input string as a base 10 number
52     flag = mpz_set_str(n, inputStr, 10);
53     assert(flag == 0);
54     // If flag is not 0 then the operation /*
55
56     // 3. Initialize the result number
57     mpz_t r;
58     mpz_init(r);
59
60     // 4. find the nth number that reversed also is prime
61     emirp(n, r);
62     mpz_out_str(stdout, 10, r);
63     printf("\n");
64     // 6. Clean up the mpz_t handles or else we will leak memory
65     mpz_clear(n);
66     return EXIT_SUCCESS;
67 }

```

In our C implementation, we used the GMP library, allowing us to replace functions in our toy language:

isPrime is replaced with the function *mpz_probab_prime_p*,

nextPrime is replaced with *mpz_nextprime*,

and *reversen* is given to us in the spec.

Other functions are also used such as:

mpz_set_ui is a function which assigns a variable (instead of `s=13`)

mpz_init is an initialising function.

mpz_sub_ui is a function which allows us to subtract.

mpz_cmp is a function which allows us to compare (instead of `r == s`)

We use these functions as we are dealing with the type *mpz_t* instead of a regular long or int. We use these functions under the assumption that they work correctly as they are library functions.