Theory of Computation Fall 2024 Problem Set 1
Amin Zolfagharian - 610300118

Problem 1.

This program increments r_2 and r_3 in each step. Once $r_1 = r_2$ the program terminates. Since in the initial configuration we have $r_2 > r_1$ and in each step r_2 increases, the program never terminates.
If we want to make sure that the program terminates it would be enough to have $r_1 \geq r_2$.

Problem 2.

(i)
```
Program: J(3, 2, NO)
         J(3, 1, YES)
         S(3)
         J(1, 1, Program)
YES:     Z(1)
         S(1)
         J(1, 1, END)
NO:      Z(1)
END
```

(ii)
```
Program: S(2)
         S(2)
         S(2)
         J(1, 2, No)
Yes:     Z(1)
         S(1)
         J(1, 1, End)
No:      Z(1)
End
```

(iii)
```
Program: S(3)
Start:   J(1, 2, Yes)
         J(1, 3, No)
         S(2)
         S(3)
         J(1, 1, Start)
Yes:     Z(1)
         S(1)
         J(1, 1, End)
No:      Z(1)
End
```

Problem 3.

Start: S(3)

J(1, 3, P)

J(1, 2, Swap)

J(1, 1, Start)

Swap: T(1, 3)

T(2, 1)

T(3, 2)

P: J(2, 5, Finish)

Z(3)

Z(4)

Add: J(3, 2, End)

S(3)

S(4)

J(3, 2, Loop)

J(1, 1, Add)

Loop: Z(4)

J(1, 1, Add)

End: T(2, 1)

T(4, 2)

J(2, 5, Finish)

Finish

In this program first we put the bigger number in R_1 and the other in R_2. Then we proceed with euclidean algorithm to compute gcd.

Problem 4.

(i) Let $f : \mathbb{N} \to \mathbb{N}$. Where $\forall x \in \mathbb{N} : f(x) = 0$. And Let $g : \mathbb{N} \to \mathbb{N}$ where $g(x, y) = y + 1$.

$$h(x, 0) = f(x)$$
$$h(x, y + 1) = g(x, h(x, y)) = h(x, y) + 1$$

It is obvious that $h(x, y) = x + y$.

(*ii*) Let f be the same as the last part. And let $g : \mathbb{N} \to \mathbb{N}$ such that $g(x, y) = x + y$.

$$h(x, 0) = f(x)$$
$$h(x, y + 1) = g(x, h(x, y)) = h(x, y) + x$$

We can see that $h(x, y) = xy$.

(*iii*) First we define the function $p(x) = x - 1$. This function can be created using recursion with $f(x) = 0$ and $g(x, y) = x$.

$$h(y) = 0$$
$$h(y + 1) = g(y, h(y)) = y$$

This function returns $x - 1$ for any $x > 0$ and returns 0 for $x = 0$. $p(x) = h(x)$.
Now we define the operation $-$:

$$x - y = \begin{cases} x - y & x \geq y \\ 0 & \text{O.W} \end{cases}$$

With $g(x, y, z) = p(z) = z - 1$ and using recursion:

$$h(x, 0) = x$$
$$h(x, y + 1) = g(x, y, h(x, y)) = h(x, y) - 1$$

With this we have $(x - y) = h(x, y)$.
We also need $sg(x)$ where:

$$sg(x) = \begin{cases} 1 & x > 0 \\ 0 & x = 0 \end{cases}$$

Using recursion:

$$h(0) = 0$$
$$h(y + 1) = g(y, h(y)) = 1$$

Also the predicate $\chi_{x>y} = sg(x - y)$.
Now with functions created above, we create the desired function:

$$h(x) = \mu z(((z + 1) * (z + 1)) > x)$$

Problem 5.

First we construct power in a recursive manner, with $g(x, y, z) = z \times x$:

$$p(x, 0) = 1$$
$$p(x, y + 1) = g(x, y, p(x, y)) = p(x, y) \times x$$
$$\implies p(x, y) = x^y$$

Now we can construct t with substitution:

$$t(x,y) = 2^x 3^y$$

Now we construct π_1 and π_2.

$$\pi_1(x) = \mu_{z<x}(\chi_{2^z|x})$$
$$\pi_2(x) = \mu_{z<x}(\chi_{3^z|x})$$

It is easy to see that $(x,y) = (\pi_1(t(x,y)), \pi_2(t(x,y)))$.

Now we define $h(x)$ such that $h(x) = 2^{f(x)}3^{f(x+1)}$, with $g(x,y) = t(\pi_2(y), \pi_1(y)+\pi_2(y))$:

$$h(0) = 2^{f(0)}3^{f(1)} = 3$$
$$h(y+1) = g(y, h(y)) = t(\pi_2(h(y)), \pi_1(h(y)) + \pi_2(h(y)))$$

Now it is easy to see that $\pi_1(h(n)) = \pi_1(2^{f(n)}3^{f(n+1)}) = f(n)$. This shows that $f(n)$ is primitive recursive.

Problem 6.

First we define a recursive function that returns the remainder of m/n. We use functions that are defined in problem 4.
Put $f(x) = 0$ and $g(x,y,z) = \chi_{(z+1)<x} \times (z+1)$.

$$h(m,0) = f(m)$$
$$h(m,n+1) = g(m,n,d(m,n)) = \chi_{h(m,n)+1<m} \times (h(m,n)+1)$$

Where χ_D is the funciton for predicate D. Now we use this function to compute d.

$$d(m,n) = 1 - sg(h(m,n))$$

Problem 7.

First we define a function that checks if a number is prime.

$$p(x) = \begin{cases} 1 & x \text{ is prime} \\ 0 & x \text{ is not prime} \end{cases}$$

Suppose $E(x,y) = \chi_{x=y}$ and let $d(m,n) = \chi_{m|n}$ from problem 6.

$$p(x) = E(x, \mu_{z\leq x}((\chi_{z>1}) \times d(z,x)))$$

This function returns 1 if smallest devisor of x, is x itself. Otherwise it returns 0.
Now we define a recursive function that for any given x returns the number of primes between 0 and x.

$$h(0) = 0$$
$$h(y+1) = g(y, h(y)) = h(y) + p(y+1)$$

And at last we define function $prime(n)$:

$$prime(n) = \mu_z(h(z) = n)$$

And for bounded minimalization we can use factorial functions, using $g(x, y) = y \times (x + 1)$:

$$\begin{gathered} f(0) = 1 \\ f(y+1) = g(y, f(y)) = f(y) \times (y+1) \\ f(x) = x! \end{gathered}$$

now we can create $prime(n)$ in a recursive manner:

$$\begin{gathered} prime(0) = 0 \\ prime(y+1) = g(y, prime(y)) = \mu_{z<prime(y)!+2}(h(z) = y+1) \end{gathered}$$

This guarantees that z searches for at least $prime(y)! + 1$ which is not devisible by first y primes. thus the $y + 1$'th prime is lesser than or equal to $prime(y)! + 1$.