Theory of Computations,
Test 1 for the course
CS 5315, Spring 2020

Name: ____ ________________________

$\frac{10}{10}$

Up to 5 handwritten pages are allowed.

1. Translate, step-by-step, the following for-loop into a primitive recursive expression:

```
int x = p + q;
for (int i = 1; i <= r; i++)
   {x = x * q;}
```

$x = 3$
$i = 1$
$x = 3 \times 2 = 6$
$i = 2$
$x = 6 \times 2 = 12$

You can use add(.,.) (sum) and mult(., .) (product) in this expression.
What is the value of this function when p = 1, q = 2, and r = 2?

$x(p, q, 0) = p + q$

$x(p, q, m+1) = x(p, q, m) * q$

$f(n_1, n_2, 0) = g(n_1, n_2)$

$f(n_1, n_2, m+1) = h(n_1, n_2, m, f(n_1, n_2, m))$

so by comparig them, we have $p = n_1$, $q = n_2$, $g = add(\pi^2_1, \pi^2_2)$

$h = mult(\pi^4_2, \pi^4_4)$

$\Rightarrow \quad x = PR(add(\pi^2_1, \pi^2_2), mult(\pi^4_2, \pi^4_4))$

$p = 1, \quad q = 2, \quad r = 2$

$x(1, 2, 0) = 1 + 2 = 3$

$x(1, 2, 1) = 3 \times 2 = 6$

$x(1, 2, 2) = 6 \times 2 = 12$

10/10

2. Translate, step-by-step, the following for-loop into a primitive recursive expression:

```
int z = p + q;
for(int i = 1; i <= r; i++)
  {for (int j = 1; j <= s; j++)
     {z = z * q;}}
```

You can use add(.,.) and mult(., .) in this expression.
What is the value of this function when p = q = r = s = 2?

$p = q = r = s = 2$

$z = p + q = 2 + 2 = 4$

$i = 1, j = 1$

$z = 4 \times 2 = 8$

$i = 1 \cdot j = 2$

$z = 8 \times 2 = 16$

$i = 2, j = 1$

$z = 16 \times 2 = 32$

$i = 2, j = 2$

$z = 32 \times 2 = 64$

$\Rightarrow \boxed{z = 64, \text{ if } p = q = r = s = 2}$

First, we consider the inner loop.

```
int z = z0
for (int j=1, j<=s; j++)
   {z = z*q;}
```

$z(z_0, q, 0) = z_0$

$z(z_0, q, m+1) = z(z_0, q, m) * q$

$f(n_1, n_2, 0) = g(n_1, n_2)$

$f(n_1, n_2, m+1) = h(n_1, n_2, m, f(n_1, n_2, m))$

$\Rightarrow n_1 = z_0, \; n_2 = q, \; k = 2, \; g = \pi^2_1 \quad h = mult(\pi^4_2, \pi^4_4)$

$\Rightarrow \boxed{aux = PR(\pi^2_1, mult(\pi^4_2, \pi^4_4))}$

Then for the outter loop.

```
int z = p+q
for (int i=1; i<=r; i++)
  { z = aux(z, q, s)
                     }
```

$\Rightarrow z(p, q, s, 0) = p + q$

$z(p, q, s, m+1) = aux(z(p, q, s, m), q, s)$

$f(n_1, n_2, n_3, 0) = g(n_1, n_2, n_3)$

$f(n_1, n_2, n_3, m+1) = h(n_1, n_2, n_3, m, f(n_1, n_2, n_3, m))$

$\Rightarrow n_1 = p. \quad n_2 = q, \quad n_3 = s$

$g = \pi^3_1 + \pi^3_2 = add(\pi^3_1, \pi^3_2)$

$h = aux(\pi^5_5, \pi^5_2, \pi^5_3)$

$\Rightarrow \boxed{z = PR(add(\pi^3_1, \pi^3_2), aux(\pi^5_5, \pi^5_2, \pi^5_3))}$

If $p = q = r = s = 2$ then

10/10

3. Translate, step-by-step, the following primitive recursive function into a for-loop:

$$f = \sigma(PR(add(\pi^2_2, \sigma(0)), add(\pi^4_1, \pi^4_3))).$$

For this function f, what is the value f(2, 0, 1)?

Let $F = \underbrace{PR(\underbrace{add(\pi^2_2, \sigma(0))}, add(\underbrace{\pi^4_1, \pi^4_3}))}$

then $f = \sigma(F)$.

from $F = PR(add(\pi^2_2, \sigma(0)), add(\pi^4_1, \pi^4_3))$

we have $k = 2$ $g(n_1, n_2) = add(\pi^2_2, \sigma(0)) = n_2 + 1$;

$h(n_1, n_2, m, F(n_1, n_2, m)) = add(\pi^4_1, \pi^4_3) = n_1 + m$.

```
int F = n2 + 1
for ( int i = 1 ; i <= m ; i++ )
{ F = n1 + i - 1 ; }

int f = F + 1
```

$$\begin{cases} F(n_1, n_2, 0) = g(n_1, n_2) \\ F(n_1, n_2, m+1) = h(n_1, n_2, m, F(n_1, n_2, m)) \end{cases}$$

$F(2, 0, 0) = \underset{n_2}{0} + 1 = 1$

$F(2, 0, 1) = \underset{n_1}{2} + \underset{m}{0} = 2$.

$f(2, 0, 1) = F(2, 0, 1) + 1 = 2 + 1 = 3$.

20/20

4-5. Prove, from scratch, that the function f(p, q) = p % (q / p) is primitive recursive. Start with the definitions of a primitive recursive function, and use only this definition in your proof -- do not simply mention results that we proved in class, prove them.

Definition: A function is called Primitive recursive if it can be obtained from 0, σ and π^k_i by using compozition and Primitive recursion.

1. add(+) is P.r.

int sum = a. $\quad f(n_1, 0) = g(n_1) \quad f(a, 0) = a$

for (int i=1, i <= b; i++) $\quad f(n_1, m+1) = h(n_1, m, f(n_1, m)) \quad f(a, m+1) = f(a, m) + 1$

{sum = sum + 1;} $\quad \Rightarrow k = 1,\ a = n_1,\ g = \pi^1_1 \quad h = \sigma(\pi^3_3)$

$\Rightarrow add = PR(\pi^1_1, \sigma \circ \pi^3_3)$, so add (+) is P.r

2, mult (*) is P.r

int mult = 0 $\quad f(n_1, 0) = g(n_1) \quad f(a, 0) = 0$

for (int i=1; i<=b; i++) $\quad f(n_1, m+1) = h(n_1, m, f(n_1, m)) \quad f(a, m+1) = f(a, m) + a$

{ mult = mult + a ;} $\quad \Rightarrow K = 1,\ a = n_1,\ g = 0.\ h = add(\pi^3_1, \pi^3_3)$

$\Rightarrow mult = PR(0, add(\pi^3_1, \pi^3_3))$, so mult (*) is P.r.

3. prev is P.r

$prev(a) = \begin{cases} 0 & \text{if } a = 0 \\ a-1 & \text{if } a = 1.2.3. \end{cases} \quad f(0) = g \quad f(m+1) = h(m, f(m)) \quad f(0) = 0. \quad f(m+1) = m$

$\Rightarrow K = 0,\ g = 0,\ h = \pi^2_1$

$\Rightarrow prev = PR(0, \pi^2_1)$, so prev is P.r.

4 sub($\dot{-}$) is P.r

int sub = a. $\quad f(n_1, 0) = g(n_1) \quad f(a, 0) = a$

for (int i=1; i<=b; i++) $\quad f(n_1, m+1) = h(n_1, m, f(n_1, m)) \quad f(a, m+1) = prev(f(a, m))$

{ sub = prev(sub);} $\quad \Rightarrow K = 1,\ a = n_1,\ g = \pi^1_1,\ h = prev \circ \pi^3_3$

$\Rightarrow sub = PR(\pi^1_1, prev \circ \pi^3_3)$, so sub($\dot{-}$) is P.r.

5. And (&&) is P.r.

s\q	0	1
0	0	0
1	0	1

so a && b = a * b, since * is P.r

so && is P.r.

next page

6. If P then Q, else R is P.r.

If P, then Q else R can be expressed as

$P * Q + (1-P) * R$ since $+$, $-$, $*$ are P.r.

so If P then Q, else R is P.r

7. eqal to 0 (eq0) is p.r.

$eq0(0)=1 \qquad f(0)=1 \qquad f(0)=g$

$eq0(m+1)=0 \qquad f(m+1)=0 \qquad f(m+1)=h(m, f(m))$

$\Rightarrow k=0, \quad g=1, \quad h=0$

$\Rightarrow eq0 = PR(\sigma \circ 0, 0)$ so eq0 is P.r.

8. equal (==) is P.r.

$a == b \Leftrightarrow (a \leq b) \&\& (b \leq a) \Leftrightarrow eq0(a \dot{-} b) \&\& eq0(b \dot{-} a)$

$\Leftrightarrow eq0(a \dot{-} b) * eq0(b \dot{-} a)$, since $*$, $\dot{-}$ and eq0 are P.r

so == is P.r.

9. rem (%) is P.r.

$rem(a, 0) = 0$

$rem(a, m+1) =$ if $(rem(a,m)+1 == a)$ return 0
else return $(rem(a,m)+1)$

since if P then Q else R, + and == are P.r so rem is P.r.

10 div (/) is P.r.

$div(a, 0) = 0.$

$div(a, m+1) =$ if $(rem(a, m+1) == 0)$ return $(div(a, m)+1)$
else return $div(a, m)$

since if P then Q else R, == and + are P.r. so div is P.r.

11. $f(p,q) = p \% (q / p)$

since we have proved that both rem(%) and div(/) are p.r. so $f(p,q)$ can be obtained from σ, 0, π^k_i, by using composition and primitive recursion, it means, $f(p,q)$ is p.r.

10/10

6. Prove that the following function f(p, q) is μ-recursive: f(p, q) = p % (q / p) when each of the values p and q is either 1 or 2, and f(p, q) is undefined for other pairs (p, q).

there are several cases.

if $p == 1$, $q == 1$ then $q/p = 1/1 = 1$, $p \% (q/p) = 1 \% 1 = 0$.

if $p == 1$, $q == 2$ then $q/p = 2/1 = 2$. $p \% (q/p) = 1 \% 2 = 1$.

if $p == 2$, $q == 1$ then $q/p = 1/2 = 0$, $p \% (q/p) = 2 \% 0$ undefined

if $p == 2$, $q == 2$ then $q/p = 2/2 = 1$, $p \% (q/p) = 2 \% 1 = 0$.

$$\text{so } f(p, q) = \begin{cases} 0, & \text{if } (p == 1 \text{ and } q == 1) \text{ or } (p == 2.\ q == 2) \\ 1, & \text{if } (p == 1 \text{ and } q == 2) \\ \text{undefined} \end{cases}$$

$$f(p, q) = \mu m.\, P([(p == 1, q == 1, m == 0) \| (p == 2, q == 2, m == 0) \| (p == 1, q == 2, m = 1)])$$

so $f(p, q)$ is μ-recursive.

10/10

7. Translate the following μ-recursive expression into a while-loop:

$$f(a, b) = \mu m.(m * a == b).$$

For this function f, what is the value of f(2, 4)? f(2, 5)?

```
int m = 0.
while ( ! (m*a == b))
    { m++ ; }
```

for f(2, 4)

int m = 0

$m*a = 0*2 = 0 \neq 4$

$m = 1$

$m*a = 1*2 = 2 \neq 4$

$m = 2$

$m*a = 2*2 == 4$ so $f(2,4) = 2 = m$.

for f(2, 5) since no m vale will meet the stop condition

so it is undefined the loop will not stop.

20/20

8-9. Suppose that someone comes up with a new proof that not every computable function is primitive recursive, by providing a new example of a function N(n) which is computable but not primitive recursive. What if, in addition to 0, π^k_i, and σ, we also allow this new function N(n) in our constructions? Let us call functions that can be obtained from 0, π^k_i, σ, and N(n) by using composition and primitive recursion *N-primitive recursive* functions. Will then every computable function be N-primitive recursive? Prove that your answer is correct.

In order to prove that there exist a computable function which is not N-primitive recursive (N-P.r), at first, we need an auxiliary notion of N-P.r. code.

We start with an expression then.

* We translate all symbols into ASCII Code for example.
  $\sigma \to$ \sigma, $\pi^k_i \to$ \pi^k_i, $0 \to$ \circ
* use ASCII Code to translate them into a binary string of 0s and 1s.
* put a 1 before the resulting binary string and interpret this string as a binary integer.

This integer is called N-P.r code.

Lemma: There exist an algorithm that given a nature number c

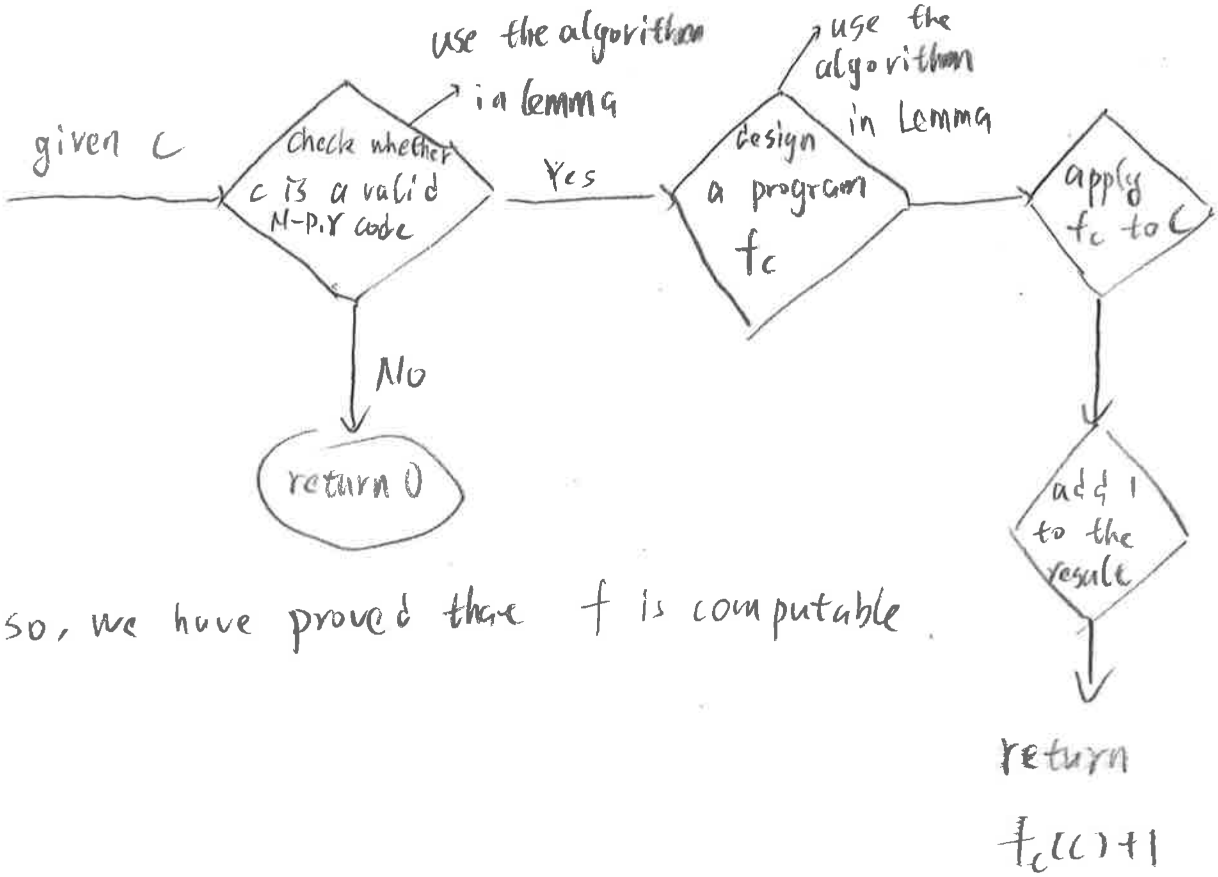
* It checks whether c is a valid N-P.r code.
* If c is a valid N-P.r code, return a Java program that computes the corresponding N-P.r function, this program will be denoted as f_c

Let's define a new function

$$f(c) = \begin{cases} f_c(c)+1 & \text{if } c \text{ is a valid N-P.r code} \\ 0 & \text{otherwise.} \end{cases}$$

Let's prove that this function is computable but not N-P.r.

First. Let's prove that f is computable



so, we have proved that f is computable.

Now, let's prove that f is not N-P.r by contradiction.

We assume that f is N-P.r. then it has a N-P.r code, we denote it by c_0. By Lemma, f_{c_0} computes the function f. this means that for every input n, $f_{c_0}(n) = f(n)$. in particular, if we let c_0 be the input, then we have $f_{c_0}(c_0) = f(c_0)$. by the definition of $f(c)$ since c_0 is a N-P.r code, we have $f_{c_0}(c_0) = f_{c_0}(c_0) + 1$, so $\cancel{f_{c_0}}(c_0) = \cancel{f_{c_0}}(c_0) + 1 \Rightarrow 0 = 1$, which is a contradiction, thus, f is not a N-P.r. function

10/10

10. Design Turing machines for computing n + 1 in unary and in binary codes.

For unary.

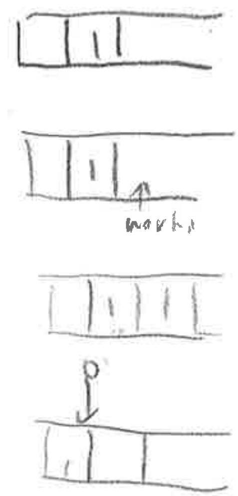
start, # → working, R.

working, 1 → R.

working, # → 1, return, L

return, 1 → L

return, # → halt



For binary

start, # → working, R.

working, 0 → 1, return, L

working, 1 → 0, right, R.

right, 1 → 0, R.

right, 0 → 1, return, L.

right, # → 1, return, L.

return, # → halt.

return, 1 → L

return, 0 → L.