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

Name: ______________________________

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;}
```

$= PR[sum(\pi^2_1, \pi^2_2), mult(\pi^4_4, \pi^4_2)]$

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?

10|10

Soln:

In mathematical terms

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

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

The general expression for $k=2$

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

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

In this case

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

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

Thus we have

$g = n_1+n_2 = Sum(\pi^2_1, \pi^2_2)$

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

$\therefore x(p,q,m) = PR[Sum(\pi^2_1, \pi^2_2), mult(\pi^4_4, \pi^4_2)]$

$\boxed{1}$ $\boxed{2}$ $\boxed{2}$

p q r

$\cancel{3}\ \cancel{6}$ 12 $\cancel{1}\ \cancel{2}$ 3

x i

The value for p = 1 q = 2 and r = 2 is 12

$\boxed{x = 12}$

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;}}
```

$= PR\left[Sum(\pi^3_1, \pi^3_2), \left(\left(PR[\pi^2_1, mult(\pi^4_4, \pi^4_2)] \right) (\pi^5_5, \pi^5_2, \pi^5_3) \right) \right]$

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

Soln:

First we breakdown the nested loop

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

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

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

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

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

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

$g = \pi^2_1$

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

$aux = PR[\pi^2_1, mult(\pi^4_4, \pi^4_2)]$

In general for $k=2$

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

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

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

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

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

$f(n_1, n_2, n_3, m+1) = aux(f(n_1, n_2, m_3, m), n_2, n_3)$

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

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

$z = PR[sum(\pi^3_1, \pi^3_2), aux(\pi^5_5, \pi^5_2, \pi^5_3)]$

In general for $k=3$

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

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

$$\therefore\ Z = PR\left[\, sum(\pi^3_1, \pi^3_2), \left(\left(PR\left[\pi^2_1, mult(\pi^4_4, \pi^4_2)\right]\right)\left(\pi^5_5, \pi^5_2, \pi^5_3\right)\right)\right]$$

$\boxed{2}$ $\boxed{2}$ $\boxed{2}$ $\boxed{2}$ $\quad \cancel{4}\ \cancel{8}\ \cancel{16}\ \cancel{32}\ 64$

$p \quad q \quad r \quad s \quad Z$

$\cancel{1}\ \cancel{2}\ 3 \quad \cancel{1}\ \cancel{2}\ 3$

$i \quad \cancel{j}$

$\cancel{1}\ \cancel{2}\ 3$

$\cancel{j}$

$$\boxed{Z(2,2,2,2) = 64}$$

$i = 0 \qquad Z = 2+2 = 4$

$i = 1 \qquad j = 1 \qquad Z = 4 \times 2 = 8$

$j = 1 \qquad j = 2 \qquad Z = 8 \times 2 = 16$

$i = 2 \qquad j = 1 \qquad Z = 16 \times 2 = 32$

$i = 2 \qquad j = 2 \qquad Z = 32 \times 2 = 64$

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)?

Soln:

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

& $f = \sigma \circ F$

⇒ The general formula.

$f(n_1, n_2 \dots n_k, 0) = g(n_1, n_2 \dots n_k)$

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

```
int f = g(n1 ... nk)
  for (int i = 1; i <= m; i++)
    { f = h(n1, ... nk, m, f); }
```

Here $k+2 = 4 \Rightarrow k = 2$

$F(n_1, n_2, 0) = g(n_1, n_2) = add(\pi^2_2, \sigma(0)) = n_2 + 1$

$F(n_1, n_2, m+1) = h(n_1, n_2, m, F(n_1, n_2, m)) = add(\pi^4_1, \pi^4_3) = n_1 + m$

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

$F(2, 0, 0) = 0 + 1 = 1$

$F(2, 0, 1) = 2 + 0 = 2$

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

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.

Soln: Definition: A function is called primitive recursive (pr) if it can be obtained from 0, 6 and Π^K_i by using composition (o) and primitive recursion (PR).

To proove $f(p,q) = p\% (q/p)$ is primitive recursive we have to proove that

1> Remainder (%) is p.r
2> Division (/) is p.r

1> Remainder is p.r.

$rem(a,0) = 0$
$$rem(a,m+1) = \begin{cases} rem(a,m)+1 & \text{if } rem(a,m)+1 < a \\ 0 & \text{otherwise} \end{cases}$$

$f(n_1,0) = g(n_1)$
$f(n_1,m+1) = h(n_1,m,f(n_1,m))$

$f(n_1,0) = 0$
$f(n_1,m+1) =$ if $f(n_1,m) < n_1$ then $f(n_1,m)+1$ else 0

Thus

$rem = PR(0,$ if $6 \circ \Pi^3_3 < \Pi^1_1$ then $6 \circ \Pi^3_3$ else $0)$

Thus remainder will be p.r if: if P(a) then f(a) else g(a) is p.r
Sum is p.r $<$ is p.r.

⇒ If P(a) then f(a) else g(a) is p.r:

Mathematically:
$$P(a) \cdot f(a) + (1 - P(a)) * g(a)$$

Proof:
if $P(a) = 1$ then $f(a)$
if $P(a) = 0$ then $g(a)$

Thus we have if P(a) then f(a) else g(a) is p.r provided P(a), f(a) and g(a) are p.r, sum is p.r, substraction is p.r, product is p.r.

⇒ Sum is p.r.

$$a+b = a+\underbrace{1+1\ldots\ldots+1}_{b\text{-times}}$$

```
int sum = a
  for (i=1; i<=b; i++)
      { sum = sum+1; }
```

$sum(a,0) = a$
$sum(a,m+1) = sum(a,m)+1$

$f(n_1,0) = g(n_1)$
$f(n_1,m+1) = h(n_1,m,f(n_1,m))$

$f(n_1,0) = n_1$
$f(n_1,m+1) = f(n_1,m)+1$

Thus $sum = PR(\Pi^1_1, S\circ\Pi^3_3)$

⇒ Prev(n) is p.r.

$Prev(0) = 0$
$prev(m+1) = m$

$f(0) = g$
$f(m+1) = h(m, f(m))$

$f(0) = 0$
$f(m+1) = m$

Thus $prev(n) = PR(0, \Pi^2_1)$

⇒ Product is p.r:

$$a*b = \underbrace{a+a+a+\ldots.a}_{b\text{-times}}$$

```
int prod = 0;
  for (int i=1; i<=b; i++)
     { prod = prod+a; }
```

$f(n_1,0) = g(n_1)$
$f(n_1,m+1) = h(n_1,m,f(n_1,m))$

$f(n_1,0) = 0$
$f(n_1,m+1) = f(n_1,m) + n_1$

Thus $prod = PR(0, sum(\Pi^3_3, \Pi^1_1))$

⇒ Substraction is p.r:

$$a \dot{-} b = a\ \underbrace{\dot{-}\,1\ \dot{-}\,1\ \ldots\ \dot{-}\,1}_{b\text{-times}}$$

```
int sub = a
  for (i=1; i<=b; i++)
     { sub = prev(sub); }
```

$sub(a,0) = a$
$sub(a,m+1) = sub(a,m) \dot{-} 1$

$f(n_1,0) = g(n_1)$
$f(n_1,m+1) = h(n_1,m,f(n,m))$

$f(n_1,0) = n_1$
$f(n_1,m+1) = f(n_1,m) \dot{-} 1 = \Pi^3_3 \dot{-} 1$

Thus $sub = PR(\Pi^1_1, prev(\Pi^3_3))$

Thus we have substraction is p.r provided prev(n) is p.r.

⇒ < is p.r:

$$a < b \iff a \dot{-} b = 0 \iff eq0(a \dot{-} b)$$

to proove < is p.r we have to proove that eq0 is p.r

⇒ eq0 is p.r:

$eq0(0) = 1$
$eq0(1) = 0$

$f(0) = 1$
$f(m+1) = 0$

$eq0 = PR(S\cdot 0, 0)$

→ Next page for 2

2> <u>Division is p.r:</u>

$$div(a,0) = 0$$
$$div(a,m+1) = \begin{cases} div(a,m)+1 & \text{if } rem(a,m+1) = 0 \\ div(a,m) & \text{otherwise} \end{cases}$$

$$f(n,0) = 0$$
$$f(n_1,m+1) = \begin{cases} \text{if } rem(n_1,m+1) == 0 \text{ then } f(n_1,m)+1 \\ \text{else } f(n_1,m) \end{cases}$$

Thus $div = PR(0, \text{ if } rem(a,m+1) == 0 \text{ then } 6 \circ \Pi^3_3 \text{ else } \Pi^3_3)$

Thus division is p.r provided remainder is p.r (which we have already prooved), if P(a) then f(a) else g(a) is p.r (which we have also already prooved), and = is p.r.

⇒ <u>= is p.r;</u>

$a = b \iff (a \le b) \,\&\&\, (b \le a)$

to proove = is p.r we have to proove && is p.r and ≤ is p.r

⇒ <u>&& is p.r:</u>

&& is product so it is p.r

⇒ <u>≤ is p.r:</u>

$a \le b \iff a \dot{-} b = 0 \iff eq\,0(a \dot{-} b)$

Thus ≤ is p.r

Hence we can say that P% (q/p) 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).

$$f(p,q) = \begin{cases} p \% (q/p) & \text{if } p \in \{1,2\} \text{ and } q \in \{1,2\} \\ \text{Undefined} & \text{otherwise} \end{cases}$$

I dont think this function would be defined for $p=2$ & $q=1$ as we would have $f(2,1) = 2\% (1/2)$ for integer division $(1/2)$ is 0 and $2\%0$ is undefined. Assuming it is $2\%0.5$ and thus equal to 0 the solution would be as given below, however we cannot have non Natural numbers like 0.5.

$$p\%(q/p) = \begin{cases} 0 & \text{if } p=1 \text{ and } q=1 \\ 0 & \text{if } p=2 \text{ and } q=1 \\ 1 & \text{if } p=1 \text{ and } q=2 \\ 0 & \text{if } p=2 \text{ and } q=2 \end{cases}$$

No need to compute

$\mu m[(p==1 \&\& q==1 \&\& m==0) || (p==2 \&\& q==1 \&\& m==0)$
$|| (p==1 \&\& q==2 \&\& m==1) || (p==2 \&\& q==2 \&\& m==0)]$

$p\%(q/p)$ calculation:

$p=1, q=1 \rightarrow 1\%(1/1) = 1\%1 = 0$

$p=2, q=1 \rightarrow 2\%(1/2) = 2\%0$ ⟶ * This should be undefined for integer division not sure how this function would be defined If we assume $(1/2) = 0.5$ & $2 \bmod 0.5 = 0$ then the result would be as shown above.

$p=1, q=2 \rightarrow 1\%(2/1) = 1\%2 = 1$

$p=2, q=2 \rightarrow 2\%(2/2) = 2\%1 = 0$

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)?

Soln:-

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

⇒ f(2, 4) ?

a = 2

b = 4

m a b

0*2 == 4 ✗ Not satisfied

1*2 == 4 ✗ Not satisfied

2*2 == 4 ✓ Satisfied (exit loop)

$\boxed{\text{Thus } f(2,4) = 2}$

⇒ f(2, 5) ?

a = 2

b = 5

0*2 == 5 ✗ Not satisfied

1*2 == 5 ✗ Not satisfied

2*2 == 5 ✗ Not satisfied

3*2 == 5 ✗ Not satisfied (loop runs continiously)

⋮

$\boxed{f(2,5) \text{ is undefined}}$

This loop never ends as no integer value of m will satisify the given condition thus f(2,5) is Undefined. This while loop will never end.

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.

Soln:

Before we start the proof we need to describe the following two things

1) Definition of a N-Pr code: we have stated that a N-pr function can be obtained from $0, \sigma, \pi^k_i$ and N(n) using composition(o) and primitive recursion thus every N-pr function can be described by an expression containing (,), o, σ, π^k_i and N(n). Now to define a N-pr code we start with this expression and then assign an integer number to this expression in following way:

- We first use latex to translate this expression into ASCII
  eg: $\sigma \to$ \sigma
  $\pi^k_i \to$ \pi^k_i
  $\circ \to$ \circ

- Now we use ASCII to transform each symbol into 0's and 1's
  eg: P → 01010000
  \ → 01011100

- Now we place 1 infront (so that we can re-construct the string back)

- Finally intrepret this binary string as an integer and this integer will be the N-pr code for the corresponding expression.

2) Lemma (w/o proof): There exists an algorithm that given a natural number c, check's whether c is a N-pr code of some N-pr expression and if yes returns a java program for computing the corresponding N-pr. expression. The java program will be denoted by f_c

How this algorithm works:

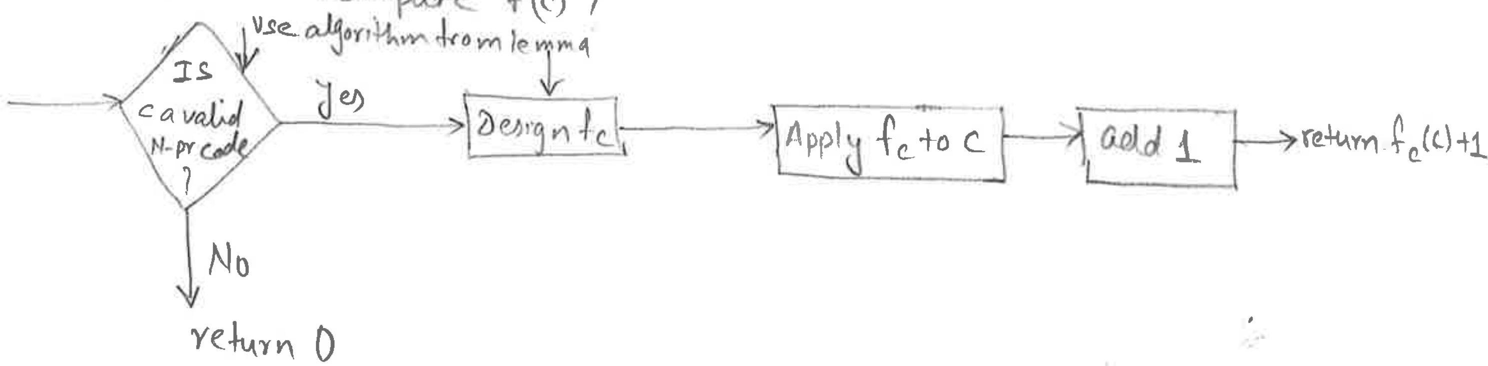
- Convert's the Natural number to binary string
- Stripp's off the first 1
- Check's whether each byte is ASCII character
- Check's whether all latex symbol's are correct
- Compiles to get f_c

⇒ Main part of the proof:

let us define the following function

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

How can we compute f(c)?



To proove that this function f(c) is not N-p.r we will use proof by contradiction.

let us assume that f(c) is N-pr and let us deduce a contradiction from this assumption

Since f(c) is N-pr it has a N-pr code, let us denote this code by c_0

for this N-pr code the algorithm from lemma will produce a Java program f_{c_0}

$$\forall c\,(f_{c_0}(c) = f(c))$$

In particular for $c = c_0$

$$f_{c_0}(c_0) = f(c_0) \quad -(\mathrm{I})$$

On the other hand, since c_0 is a valid N-pr code from the definition of function $f(c)$, we have

$$f(c_0) = f_{c_0}(c_0) + 1 \quad -(\mathrm{II})$$

from (I) & (II) we have

$$\cancel{f_{c_0}(c_0)} = \cancel{f_{c_0}(c_0)} + 1$$

$$0 = 1$$

Thus we have a contradiction which means that our assumption that this function is N-pr was wrong hence we conclude that $f(c)$ is not N-pr.

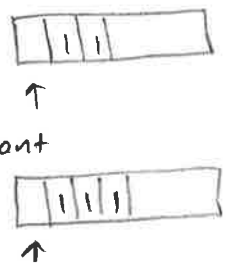
8/10

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

⇒ In Unary:

we have

we want



Directives

Start # → working, R

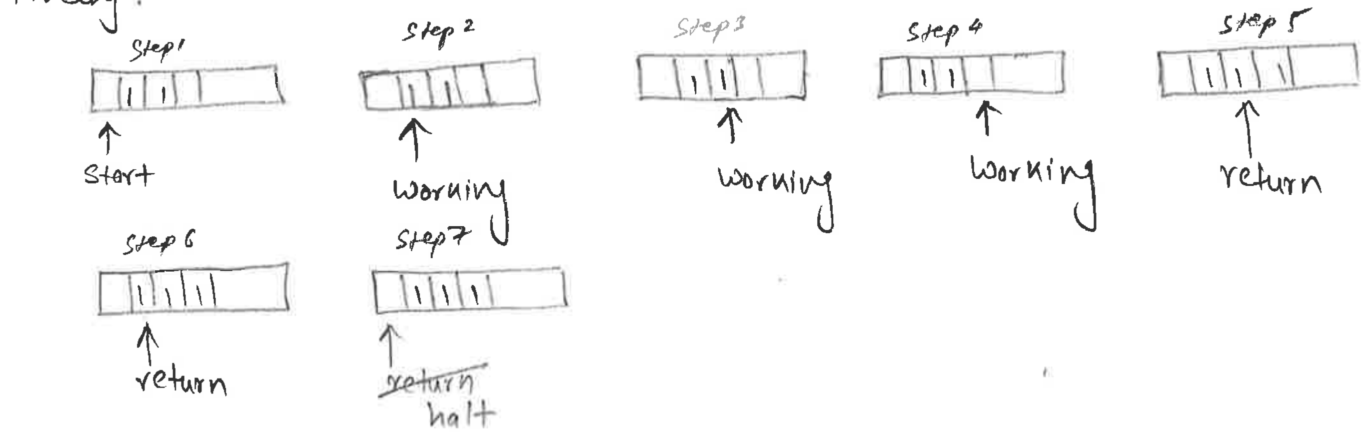
working, 1 → R

working, # → 1, Return, L

Return, 1 → L

Return, # → halt

Tracing:



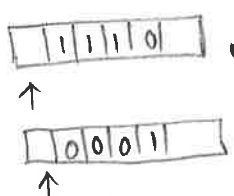
⇒ In binary:

Motivation: In binary addition of 1 means we move from least significant bit to most significant bit changing all the 1's to 0's until we get a 0 or blank and we change that 0 to 1 and return.

Also in turing machine the least significant bit is stored first i.e. Number's are stored in reverse order.

7 we have

8 we want



Directives: working, # → 1, return L.

Start, # → R, working

working, 1 → 0, R

working, 0 → 1, Return, L

Return, 0/1 → L

Return, # → halt

Tracing: