Theory of Computations,
Test 1 for the course
CS 5315, Spring 2019

Name:

Up to 5 handwritten pages are allowed.

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

```c
int x = a * b;
for (int i = 1; i <= c; i++)
  {x = x + b;}
```

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

\[
\text{mathematical terms: } \chi(a, b, 0) = a \times b \\
\chi(a, b, m+1) = \chi(a, b, m) + b
\]

\[
\text{renaming: } f(n_1, n_2, 0) = n_1 \times n_2 \\
f(n_1, n_2, m+1) = f(n_1, n_2, m) + n_2
\]

In general, for a primitive recursive function of k variables,

\[
f(n_1, \ldots, n_k, 0) = g(n_1, \ldots, n_k) \\
f(n_1, \ldots, n_k, m+1) = h(n_1, \ldots, n_k, m, f(n_1, \ldots, n_k, m))
\]

Here, \( k = 2 \),

\[
f(n_1, n_2, 0) = g(n_1, n_2) = n_1 \times n_2 = \text{mult}(\pi^{2}_1, \pi^{2}_2) \\
f(n_1, n_2, m+1) = h(n_1, n_2, m, f(n_1, n_2, m)) \\
= f(n_1, n_2, m) + n_2 = \text{add}(\pi^{4}_1, \pi^{4}_2)
\]

Final expression:

\[
\chi = \text{PR} \left( \text{mult}(\pi^{2}_1, \pi^{2}_2), \text{add}(\pi^{4}_1, \pi^{4}_2) \right)
\]

\( a = 2 \), \( b = 1 \), \( c = 2 \) \( \Rightarrow \)

\[
\chi(2, 1, 0) = 2 \times 1 = 2 \\
\chi(2, 1, 1) = 2 + 1 = 3 \\
\chi(2, 1, 2) = 3 + 1 = 4
\]

The value of the function is 4 when \( a = 2, b = 1, c = 2 \).
2. Translate, step-by-step, the following for-loop into a primitive recursive expression:

```c
int z = a + b;
for(int i = 1; i <= a; i++)
    for (int j = 1; j <= c; j++)
        z = b + z;
```

You can use add(...) and mult(...) in this expression.
What is the value of this function when \(a = 1, b = 2,\) and \(c = 2?\)

```c
int z = z₀ , where z₀ is the initial value.

for (j=1; j<=c ; j++)
    z = b + z;

z (z₀, b, 0) = z₀
z (z₀, b, m+1)=b+z (z₀, b, m)
```

\[f(n₁, n₂, 0) = g(n₁, n₂) = n₁ = \pi₁^2\]

\[f(n₁, n₂, m+1) = h(n₁, n₂, m, f(n₁, n₂, m)) = n₂ + f(n₁, n₂, m) = \text{add}(\pi₂^4, \pi₄^4)\]

\[\text{aux} = \text{PR}(\pi₁^2, \text{add}(\pi₂^4, \pi₄^4))\]

\[z(a, b, c, 0) = a \times b\]

\[z(a, b, c, m+1) = \text{aux}(z(a, b, c, m), b, c)\]

\[f(n₁, n₂, n₃, 0) = n₁ \times n₂ = \text{mult}(\pi₃^3, \pi₂^3)\]

\[f(n₁, n₂, n₃, m+1) = \text{aux}(f(n₁, n₂, n₃, m), n₂, n₃)\]

\[h(n₁, n₂, n₃, m, f(n₁, n₂, n₃, m)) = \text{aux}(\pi₃^5, \pi₂^5, \pi₃^5)\]

**Final expression**

\[z = \text{PR}(\text{mult}(\pi₁^3, \pi₂^3), \text{aux}(\pi₃^5, \pi₂^5, \pi₃^5))\]

\[a = 1, b = 2, c = 2\]

\[j=1, z = \frac{b}{2} + (1 \times 2) = 4\]

\[j=2, z = 2 + 4 = 6\]

**Ans. 6.**
Translate, step-by-step, the following primitive recursive function into a for-loop:

\[ f = \sigma(\sigma(\text{PR}(\text{add}(\pi^2_1, \pi^2_2), \text{add}(\pi^4_1, \pi^4_2, \pi^4_3)))) \]

For this function \( f \), what is the value \( f(2, 3, 1) \)? Provide an explicit formula for the corresponding function.

Let \( H = \text{PR}(\text{add}(\pi^2_1, \pi^2_2), \text{add}(\pi^4_1, \pi^4_2, \pi^4_3)) \)

Let, \( F = \sigma \circ H \)

\( f = \sigma \circ F \)

\[ H(n_1, n_2, \ldots, n_k, 0) = g(n_1, n_2, \ldots, n_k) \]

\[ H(n_1, n_2, \ldots, n_k, m+1) = h(n_1, \ldots, n_k, m, H(n_1, \ldots, n_2, m)) \]

\( k = 2 \)

\[ H(n_1, n_2, 0) = g(n_1, n_2) = n_1 + n_2 \]

\[ H(n_1, n_2, m+1) = h(n_1, n_2, m, H(n_1, n_2, m)) = n_1 + n_2 + m \]

**Code:**

```plaintext
int H = n1 + n2

for (int i = 1; i <= m; i++)

{ H = n1 + n2 + (i-1); }

int F = H + 1

int f = F + 1;
```

\( f(2, 3, 1) \): \[ H(2, 3, 0) = 2 + 3 = 5 \]

\[ H(2, 3, 1) = 2 + 3 + 0 = 5 \]

\[ F(2, 3, 1) = 5 + 1 = 6 \]

\& \( f(2, 3, 1) = 6 + 1 = 7 \)

**Explicit formula for the function**

\( f(x, y, z) = x + y + \frac{z(z-1)}{2} + 2 \)
4-5. Prove, from scratch, that the function \( f(a, b) = a \div (a \% b) \) 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.

\[
\begin{align*}
div(a, 0) &= 0 \\
div(a, m+1) &= \begin{cases} 
\text{div}(a, m) + 1 & \text{if } \text{rem}(a, m+1) = 0 \\
\text{else } \text{div}(a, m) & 
\end{cases}
\end{align*}
\]

To prove \( a \div (a \% b) \) is p.r. we need to prove remainder is p.r.

\[
\begin{align*}
\text{rem}(a, 0) &= 0 \\
\text{rem}(a, m+1) &= \begin{cases} 
\text{rem}(a, m) + 1 & \text{if } \text{rem}(a, m+1) = 0 \\
0, \text{ otherwise.} & 
\end{cases}
\end{align*}
\]

In order to prove \( \text{rem} \) p.r., we need to prove

(a) \text{if-else-then } p.r.,

(b) \( < \) is p.r.

\[
\text{if-else-then } p.r.
\]

\[
\text{if } f(a, m+1) < a \text{ then } f(a, m) + 1 \\
\text{else } 0
\]

Let \( p = f(a, m+1) < a \)

\[g = f(a, m) + 1\]

\[h = 0\]
we get,

\[ p(n) = q(n) + (1 - p(n)) + h(n) \]

addition

\[ a + b = a + 1 + \ldots + 1 \]

\[ f(m, 0) = m \]

\[ f(m, m + 1) = f(m, m) + 1 \]

\[ f(m, 0) = f(m) = \tau^1 \]

\[ f(m, m + 1) = h = 6 \circ \tau_3^3 \]

\[ \text{add} = \text{PR} \left( \tau^1, \ 6 \circ \tau_3^3 \right) \]

subtraction

\[ a - b = a - 1 - \ldots - 1 \]

\[ \text{sub} (a - 0) = a \]

\[ \text{sub} (a, m + 1) = \text{prev} \ (\text{sub} (a, m)) \]

\[ \text{prev} \quad \text{if} \quad \text{prev} (m) = \begin{cases} m - 1, & \text{if } m \geq 1 \\ 0, & \text{otherwise} \end{cases} \]

\[ \text{prev} = \text{PR} \ (0, \ \tau_3^2) \]

\[ \text{sub} \text{ is } \text{PR}, \text{ as } \text{prev} \text{ is } \text{PR} \]
\[ a \times b = a + \ldots + a \]

\[ \text{mult} \]

\[ \text{mult} (a, 0) = 0 \]

\[ \text{mult} (a, m+1) = \text{mult} (a, m) + a \]

\[ f(n, 0) = 0 \]

\[ f(n, m+1) = f(n, m) + n \]

\[ \Theta = 0 \]

\[ n = \text{sum} (\Pi_3^3, \Pi_1^3) \]

\[ \text{mult} = \text{PR} (0, \text{sum} (\Pi_3^3, \Pi_1^3)) \]

\[ \text{mult} = \text{PR} (0, (\text{PR} (\Pi_1^3, \text{sum} \Pi_3^3))) (\Pi_3^3, \Pi_1^3) \]

\[ (f(a, m+1) < a) \times (f(a, m) + 1) + (1 - f(a, m+1) < a) \times 0 \]

\[ \text{not is PR} : \]

\[ \text{not} (0) = 1 \]

\[ \text{not} (1) = 0 \]

\[ n = 1 - n \]

\[ \text{not} n = \text{not} (\text{PR}) = 0 \]

\[ \text{PR} \]

\[ \text{And} \]

\[ \begin{array}{c|c|c|c|c}
  a & b & 0 & 1 \\
  \\hline
  0 & 0 & 0 & 0 \\
  1 & 0 & 0 & 1 \\
\end{array} \]

\[ \text{or} \]

\[ \text{mult is PR} \]

\[ \text{PR is PR} \]
\[
\begin{array}{c|c|c}
0 & 0 & 1 \\
0 & 0 & 1 \\
1 & 1 & 1 \\
\end{array}
\]
on (11) : 0 0 1 on is odd.

\[! (a > b) \equiv P.R.\]
so, \[\leq\] is P.R.

\[a = b \iff a \leq b 8 8 8 6 \leq a\]
equal to zero:
\[eq + 0 \in \mathbb{C}(m) = \begin{cases}
\text{True, if } \exists n \\
\text{False, otherwise}
\end{cases}
\]
\[eq + 0 \in \mathbb{C}(0) = 1, \quad f(0) = 1\]
\[eq + 0 \in \mathbb{C}(m+1) = 0, \quad f(m+1) = 0\]
P.R. (0, 0, 0)
a = 0, a = b \quad a = b = 0 \iff a \leq b
0 \leq b \iff eq + 0 \in \mathbb{C}(a = b)
0 = b \iff a \leq b 8 8 8 \leq a

\[a \neq b \iff ! (a = b)\]
is P.R. so \[\neq\] is P.R.
\[\therefore a < b \iff a \leq b 8 8 8 ! a = b\]
< is P.R as all are P.R.
so the function is P.R. (proven)
6. Prove that the following function f(a, b) is μ-recursive: \( f(a, b) = (\neg a) \&\& (\neg b) \) when each of the values a and b is either 0 or 1, and f(a, b) is undefined for other pairs (a, b).

\[
f(a, b) = \begin{cases} 
    a == 0 \&\& b == 0, & m == 1 \\
    a == 0 \&\& b == 1, & m == 0 \\
    a == 1 \&\& b == 0, & m == 0 \\
    a == 1 \&\& b == 1, & m == 0 \\
    \text{undefined} & \text{otherwise}
\end{cases}
\]

\( \mu m. ( ( m == (\neg a) \&\& (\neg b) ) \&\& (a == 0 \| a == 1) \&\& (b == 0 \| b == 1) ) \)

\( \mu m. ( (a == 0 \&\& b == 0 \&\& m == 1) \| \\
    (a == 0 \&\& b == 1 \&\& m == 0) \| \\
    (a == 1 \&\& b == 0 \&\& m == 0) \| \\
    (a == 1 \&\& b == 1 \&\& m == 0) ) 
\)

\( \text{int } m = 0 \)

\( \text{while } ( \neg (m == (\neg a) \&\& (\neg b)) \&\& (a == 0 \| a == 1) \&\& (b == 0 \| b == 1) ) \)

\( \{ m++ ; \} \)
7. Translate the following μ-recursive expression into a while-loop:
\[ f(a, b) = \mu m. (m * a \geq b). \]
For this function \( f \), what is the value of \( f(2, 5) \)?

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

\[ f(2, 5) = \]

\[ \underline{3} \]

\[ 2 \quad 5 \quad 0 \]

\[ a \quad b \quad m \]

The value of \( f(2, 5) = 3 \)
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, 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, 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 exists a computable function which is not _N-primitive recursive_, at first we need some auxiliary notion which is the code of a primitive recursive function that can be obtained from 0, π^k, σ, and N by using a composition and _N primitive function_.

Therefore, every function can be described by an expression containing (,), 0, 0, π^k, and N(n).

In LaTeX, the following are the expressions to translate into ASCII:

- we replace σ with \sigma
- we replace π^k with \pi \wedge k - 1
- we replace σ with \sigma

Then we use ASCII to translate sequence into 0's and 1's, we append 1 in front and interpreted it as a binary integer. This integer is called the code of _N-primitive recursive function_.

**Lemma:** There exist an algorithm:

* given a natural e, checks whether e is a code of a _N-primitive recursive function_ and if yes, returns the executable
file compiling the value of this \( N \)-primitive function. The file will be denoted as \( f_c \).

Now we will build a function that is computable but not \( N \)-p.r.p.

Let's construct a func. \( f(e) \) about which we will prove that it is computable but not \( N \)-p.r.p.

Defining function: \( f(e) = \begin{cases} f_c(e) + 1, & \text{if } e \text{ is a code of the } N \text{-p.r.p.} \\ 0, & \text{otherwise} \end{cases} \)

First prove \( f(e) \) is computable.

The above algorithm shows that if \( C \) is a code of \( N \)-p.r.p., then it will produce \( f_c \), we run \( f_c \) and we use the input, finally we add 1 to the result of \( f_c(e) \), so we
get $f(e) + 1$

Now we will prove by contradiction that $f(e)$ is not p.r.

Assuming $f$ is a $\text{N-}p.r.$, it has a code, let's denote the code as $c_0$.

$$f_{c_0}(n) = f(n)$$

In particular $n = c_0$

so, $f_{c_0}(c_0) = f(c_0)$

As $c_0$ is a code, we get,

$$f(c_0) = f_{c_0}(c_0) + 1
\text{ and } f_{c_0}(c_0) + 1
= 0 + 1 \text{ which is a contradiction.}$$

$\therefore f(e)$ is computable but not $\text{N-primitive recursive}$

(proved)