ASSIGNMENT-11

A0223677B

1A:- $h(x) = n \leq \sqrt{2}x < n+1$

By using MATHEMATICAL INDUCTION:-

for $h(x)$ to be PRIMITIVE RECURSIVE, it should be COMPUTABLE.

Base case (i):-

for $n=1$

As:- $n \leq \sqrt{2}x < n+1$

Square on all parts

$\Rightarrow n^2 \leq 2x^2 < (n+1)^2$

Put $n=1$

$\Rightarrow (1)^2 \leq 2x^2 < (1+1)^2$

$\Rightarrow 1 \leq 2x^2 < 4$

$\Rightarrow \frac{1}{2} \leq x^2 < 2$

$\therefore$ "x" is COMPUTABLE for x in between $\frac{1}{\sqrt{2}}$ and $\sqrt{2}$

$\therefore$ $h(x)$ is COMPUTABLE.

Now for $n=2$:-

$n^2 \leq 2x^2 < (n+1)^2$

$\Rightarrow (2)^2 \leq 2x^2 < (2+1)^2$

$\Rightarrow 4 \leq 2x^2 < 9$

$\Rightarrow 2 \leq x^2 < \frac{9}{2}$

$\therefore$ "x" is COMPUTABLE for x in between $\sqrt{2}$ and $\frac{3}{\sqrt{2}}$.

$\therefore$ $h(x)$ is COMPUTABLE.

Now:-

As, $\boxed{h(x) = n \leq \sqrt{2}x < n+1}$

By Induction; $c = \sqrt{2}$

∴ This equation can be reformed as;

$h(x) = n \leq cx < n+1$

Hence,

For every $n \in \mathbb{N}$, there exists $h(x) = n \leq cx < n+1$ that is COMPUTABILE.

∴ $h(x)$ is COMPUTABLE.

We know that every primitive recursive function is COMPUTABLE...

Re-writing equation :-

$n < cx$ (OR) $n = cx$ AND $n+1 > cx$

"c" : constant "c" : constant

$\Rightarrow \quad n < cx \mid n = cx \wedge n+1 > cx$

As $\mid$ (OR), $\wedge$ are PRIMITIVE RECURSIVE,

we can say that $h(x)$ is also PRIMITIVE RECURSIVE.

2A:- LCM (least Common Multiple)

$lcm(x,y)$ ≡ LCM of x and y

≡ Smallest number that is multiple of both x and y.

Lets suppose,

LCM of x and y be L, and this L satisfies both $\frac{x}{n}$ and $\frac{y}{n}$.

$\therefore lcm(x,y) = \min_{W \leq x*y} [(x|W) \wedge (y|L)]$ → (i)

$\because$ L is LCM of x and y

$x|L$ and $y|L$ ≡ x is divisible by L and y is divisible by L.

In equation (i) :-

All the three operations ($|$, $\wedge$ and $*$) are PRIMITIVE RECURSIVE.

Hence, LCM is evidently PRIMITIVE RECURSIVE.

GCD (Greatest Common Divisor)

$gcd(x,y)$ ≡ GCD of x and y

≡ Largest/greatest common divisor of both x and y.

We Know :- $\boxed{(LCM) * (GCD) = x * y}$

$\Rightarrow$ $\boxed{GCD = \frac{x*y}{LCM} \rightarrow (ii)}$

We Know from (ii) :-

$/$, $*$ are PRIMITIVE RECURSIVE

∴ Evidently, GCD is also PRIMITIVE RECURSIVE.

3A:-

Yes, there is a computable predicate $P(x_1, x_2, \ldots, x_n, y)$ such that $\min_y P(x_1, x_2, \ldots x_n, y)$ is NOT-COMPUTABLE.

There are two cases for the predicate: $P(x_1, x_2, \ldots, x_n, y)$

CASE (i) :- If the predicate, $P(x_1, x_2, \ldots, x_n, y)$ is a true value then, $\min_y P(x_1, x_2, \ldots x_n, y)$ is COMPUTABLE. ≡ [$\min_y$ ≡ Min value of y]

CASE (ii) :- But, what if the predicate $P(x_1, x_2, \ldots x_n, y)$ is a False value.

In this case, $\min_y$ is not going to be COMPUTABLE.

∴ From case (ii), we can apparently be sure that, there exists a computable predicate $P(x_1, x_2, \ldots x_n, y)$ such that $\min_y P(x_1, x_2, \ldots, x_n, y)$ is NOT-COMPUTABLE.

3pA:-

$f(x_1, x_2, \ldots x_n)$ = Function of "n" variables.

$f'(\langle x_1, x_2, \ldots x_n \rangle) = f(x_1, x_2, \ldots x_n)$ for all $x_1, x_2 \ldots x_n$

Show:-

f' is PARTIALLY COMPUTABLE $\Leftrightarrow$ f is PARTIALLY COMPUTABLE

There are two cases here:-

CASE (i):-

If f' is partially computable, then prove that f is partially computable.

Now:- We know that,

f' is PARTIALLY COMPUTABLE $\Rightarrow$

$f'(\langle x_1, x_2, \ldots x_n \rangle)$ is PARTIALLY COMPUTABLE.

To prove:- f to be PARTIALLY COMPUTABLE,

We should show that $f(x_1, x_2, \ldots x_n)$ to be PARTIALLY COMPUTABLE.

Now,

As $f'(\langle x_1, x_2, \ldots x_n \rangle)$ is partially computable, there exists an L-program to compute this. $\rightarrow$(i)

Break this list into n parameters (U can use a L-program to do this)...

Now that the list $\equiv f'(\langle x_1, x_2, \ldots x_n \rangle)$ is broken into n-parameters $f(x_1, x_2, \ldots x_n)$. We can now use the L-program

used in (i) i.e; the L-program used to compute $f'(|x_1, x_2, \cdots x_n|)$ to compute $\underbrace{f(x_1, x_2, \cdots x_n)}_{\equiv \text{Broken/separated n-parameters.}}$

∴ If $f'(|x_1, x_2, \cdots x_n|)$ is PARTIALLY COMPUTABLE then $f(x_1, x_2, \cdots x_n)$ is PARTIALLY COMPUTABLE.

CASE(ii):-

If f is PARTIALLY COMPUTABLE, then f' is PARTIALLY COMPUTABLE.

NOW:-

As f is PARTIALLY COMPUTABLE i.e; $f(x_1, x_2, \cdots x_n)$ is PARTIALLY COMPUTABLE, implies there exists a L-program to compute this → (ii).

Now, $f'(|x_1, x_2, \cdots x_n|)$ [To compute this]

We have to do the following:

Step(i): Firstly, convert $f(x_1, x_2, \cdots x_n)$ into a list of n-values i.e; $|x_1, x_2, \cdots x_n|$ using an appropriate L-program.

Step(ii): Secondly, use the L-program mentioned above in (ii), i.e; the L-program used to compute $f(x_1, x_2, \cdots x_n)$, to compute the function of list of values(n) we've got in step(i).

∴ $f'(|x_1, x_2, \cdots x_n|)$ is PARTIALLY COMPUTABLE now.

∴ $f'(|x_1, x_2, \cdots x_n|)$ is PARTIALLY COMPUTABLE if and only if, $f(x_1, x_2, \cdots x_n)$ is PARTIALLY COMPUTABLE.

REFERENCE = Textbook

5A:- Given :-

Sort $([x_1, x_2, \cdots x_n]) = [y_1, y_2, \cdots y_n]$ where $y_1, y_2, \cdots y_n$ is a permutation of $x_1, x_2, \cdots x_n$ such that $y_1 \le y_2 \le \cdots \le y_n$

Show that :-

Sort(x) is PRIMITIVE RECURSIVE.

Let's define the predicate, perm(q) which means that

$q = [q_1, q_2, \cdots q_k] \equiv$ Permutation of $[1, 2, \cdots k]$

$$\therefore \boxed{\text{perm}(q) = (\forall i)_{1 \le i \le \text{size}(q)} (\exists j)_{1 \le j \le \text{size}(q)} [\text{item}(q, j) = i]}$$

$\therefore$ perm is PRIMITIVE RECURSIVE

Now :-

$$\text{Sort}(n) = (\min t)_{t \le \text{list}(n,n)} [\text{size}(t) = \text{size}(n) \text{ and } \text{sorted}(t) \text{ and } \text{perm}_2(t, n)],$$

Here :- sorted(t) $\equiv$ predicate [t is sorted]

$\text{perm}_2(t, n) \equiv$ predicate [t is a permutation of n]

$$\therefore \boxed{\text{sorted}(t) = (\forall i)_{1 \le i \le \text{size}(t) - 1} [\text{item}(t, i) \le \text{item}(t, i+1)]}$$

(and)
$$\boxed{\text{perm}_2(t, n) = (\exists q)_{q \le \text{list}(\text{size}(n), \text{size}(n))} [\text{perm}(q) \text{ and } \text{size}(q) = \text{size}(n) \text{ and } (\forall i)_{1 \le i \le \text{size}(n)} [\text{item}(t, i) = \text{item}(n, \text{item}(q, i))]]}$$

$\therefore$ Both sorted and perm_2 are PRIMITIVE RECURSIVE.

$\therefore$ Sort is PRIMITIVE RECURSIVE.