Rui Ji (1000340918)

1. Following is a RM program which computes the function $f(x) = 2x$, suppose x is stored at R_1.

$$\begin{array}{ll} c_0 : R_2 \leftarrow 0 & Z_2 \\ c_1 : goto\ 6\ if\ R_1 = R_2 & J_{1,2,6} \\ c_2 : R_1 \leftarrow R_1 + 1 & S_1 \\ c_3 : R_2 \leftarrow R_2 + 1 & S_2 \\ c_4 : R_2 \leftarrow R_2 + 1 & S_2 \\ c_5 : goto\ 1\ if\ R_1 = R_1 & J_{1,1,1} \\ c_6 & \end{array}$$

First we initialize R_2 to be 0. Command c_1 to c_5 is a loop, within the loop we add 2 to R_2 each time and add 1 R_1, and we will return R_1 when $R_1 = R_2$.
Note that suppose after n_{th} iteration $R_1 = R_2$, which means $n + x = 2n$ then $n = x \Rightarrow R_1 = 2x$.

□

2. (a) Since limited subtraction, bounded sum, bounded products and divisibility are all primitive recursive,

$$Bit(x,i) = 2^{i+1} | (x \dot{-} \sum_{t<i} 2^t B(x,t))$$

Note

$$2^{i+1} = h(2, i+1) = \prod_{z<i+1} 2$$

$$2^t = h(2,t) = \prod_{z<t} 2$$

then $Bit(x,i)$ is also primitive recursive.
Following is an example computes $Bit(6,0), Bit(6,1), Bit(6,2)$,

$$Bit(6,0) = 2^{0+1} | (6 \dot{-} \sum_{t<0} 2^t B(6,t))$$

by definition of bound sum we know that

$$\sum_{t<0} 2^t B(6,t) = 0$$

Then

$$Bit(6,0) = 2 | (6 - 0) = 0$$

$$Bit(6,1) = 2^{1+1} | (6 \dot{-} \sum_{t<1} 2^t B(6,t))$$

by definition of bound sum we know that

$$\sum_{t<1} 2^t B(6,t) = B(6,0) = 0$$

Then

$$Bit(6,1) = 4 | (6 - 0) = 1$$

$$Bit(6,2) = 2^{2+1} | (6 \dot{-} \sum_{t<2} 2^t B(6,t))$$

by definition of bound sum we know that

$$\sum_{t<1} 2^t B(6,t) = 2^1 B(6,1) = 2$$

Then

$$Bit(6,2) = 8|(6-2) = 0$$

(b) Suppose x in binary has n bits, we know that $2^{(x-1)} \geq x$ for $(x \in N \wedge 1 \leq x)$, then i, where $i < x$ cover all n bits of x.
Since bounded sum and $Bit(x,i)$ is primitive recursive,

$$NumOnes(x) = \sum_{i<x} B(x,i)$$

then $NumOnes(x)$ is also primitive recursive.

□

3. (a) Claim: A is neither recursive nor r.e. A^c is r.e but not recursive.
First we show that A is not r.e. Notice, it suffices to show that

$$K^c \leq_m A$$

Thus we want a total computable function $f(x)$ such that

$$x \in K^c \Leftrightarrow f(x) \in A$$

i.e we want

$$\{x\}_1(x) = \infty \Leftrightarrow dom(\{f(x)\}_1) \subseteq PRIMES$$

We can define $f(x)$ implicitly using the S-m-n Theorem as follows:

$$\{f(x)\}_1(y) = \begin{cases} \{x\}_1(x) & \text{if } y \neq 2 \\ y & \text{if } y = 2 \end{cases}$$

Thus if $\{x\}_1(x)$ is defined, then $\{f(x)\}_1(y)$ is only defined for all $y \in N$, so $dom(\{f(x)\}_1) \not\subseteq PRIMES$.
But if $\{x\}_1(x)$ is undefined then $\{f(x)\}_1(y)$ is undefined for only for $y = 2$; hence, $dom(\{f(x)\}_1) \subseteq PRIMES$.

Now we we want to show that A^c is is r.e.

$$A^c = \{x | dom(\{x\}_1) \not\subseteq PRIMES\}$$

Notice that $x \in A^c$ iff there is some input u and some v such that v codes a halting computation of program $\{x\}$ on input u, and u is not a prime. Using the T-predicate, we have,

$$x \in A^c \leftrightarrow \exists u \; \exists v \; [T(x,u,v) \wedge \neg Prime(u)]$$

using a pairing function to combine both existential quantifiers into one quantifier,

$$x \in A^c \leftrightarrow \exists z\ [T(x, K(z), L(z)) \wedge \neg Prime(K(z))]$$

We get the form $x \in A^c \leftrightarrow \exists z\ R(x,z)$ where R is recursive. Hence A^c is r.e.

It follows that A is not recursive, and hence A^c is not recursive. It also follows that A is not r.e., because otherwise A would be recursive.

(b) CLAIM: B is r.e but not recursive, B^c is neither recursive nor r.e.

First we show that B is r.e. Let's define following function $UP(z,x)$

$$UP(z,x) = U(min\ y < (A(z_0,x)+z)\ T(z_1,x,y))$$

- It is clear that $UP(z,x)$ is a total function, since U, min, A, T are all total function.
- Also for each $e \in N$, the unary function $UP(e,x)$ is primitive recursive, since U, A, T are primitive recursive function and $min\ y < (A(z_0,x)+z)$ is bounded for each x; hence, $UP(e,x)$ is primitive recursive.
- For each unary primitive recursive function $f(x)$, there exist *Ackermann's Function* $A_n(x)$, such that $A(n,x)+B > f(x)$; Hence, exist some k such that $A(n,x)+B > Comp_{\mathrm{P}}(x)$, where P computes f.

  Every primitive recursive function $f(\vec{x})$ is computable by a RM program P such that the function $Comp_{\mathrm{P}}(\vec{x})$ is primitive recursive, where

  $$Comp_{\mathrm{P}}(\vec{x})\ is\ the\ number\ coding\ the\ computation\ of\ P\ on\ input\ \vec{x}$$

  using KNFT we have,

  $$f(x) = U(min\ y < (A(k,x)+B)\ T(\#\mathrm{P},x,y))$$

  let $z_0 = max(A,B)$, $z_1 = \#\mathrm{P}$ and $z = 2^{z_0}2^{z_1}$, we have

  $$f(x) = U(min\ y < (A(z_0,x)+z)\ T(z_1,x,y))$$

  hence, for each unary primitive recursive function $f(x)$, there exist some $e \in N$ such that $f(x) = UP(e,x)$.

Then we have $B = ran(UP)$, which means B is r.e.

Now we show that B is not recursive.

Intuitively $B \neq N$, then B is not recursive It follows that B is not recursive, and hence B^c is not recursive. It also follows that B^c is not r.e., because otherwise B would be recursive.

4. (a) Let #P be the number encoding the RM program P.

$$STATE_{\mathrm{P}}(\vec{x},0) = g(\vec{x}) = p_0^0 p_1^{x-1} \ldots p_n^{x_n}$$
$$STATE_{\mathrm{P}}(\vec{x},t+1) = Nex(STATE_{\mathrm{P}}(\vec{x},t), \#\mathrm{P})$$

Note $g(\vec{x})$ is primitive recursive; hence, $g(\vec{x}) \in \varepsilon$, also we assuming that $Nex(u,z) \in \varepsilon$, where $Nex(u,z) = u$ if u is the halting states, then we can get $STATE_{\mathrm{P}}(\vec{x},t)$ is also in ε.

(b) For any $f \in \mathcal{C}$, suppose program p computes f, by definition we know that there exist $k \in N$ such that $Time_{\text{P}}(\vec{x}) \leq E_k(x_1+x_2+...+x_n)$, which means $f(x)$ will halt within $E_k(x_1+x_2+...+x_n)$ steps.

Using KNFT we can get $f(\vec{x}) = U(min\ y < E_k(x_1 + x_2 + ... + x_n) \quad T(\#\text{P}, \vec{x}, STATE_{\text{P}}(\vec{x}, y)))$, where$T(x, y, z)$ is Kleene T predicate.

Since U, T, min are all primitive recursive; hence, $U, T, min \in \varepsilon$, also we know that $E_k \in \varepsilon$; therefore, $f \in \varepsilon$. Then we get $\mathcal{C} \in \varepsilon$.

□