Due: Friday, November 20, beginning of tutorial

1. Write a register machine program which computes the function $f(x) = 2x$. Be sure to respect our input/output conventions for register machines. Write your program in the style of the Copy program given on page 55 of the Notes. DO NOT USE MACROS. (Write your program in the original RM machine language.) Explain your program so that the marker can easily understand it.

   Solution:
   Initially the input x is in register R_1, and all other registers are 0. Our program first copies R_1 into R_2, and then alternately increments R_3 and R_1 until $R_3 = R_2$. This has the effect of adding R_2 to R_1, so that the output register R_1 winds up with the value $2x$.

   COMMENT: The first 3 commands copy R_1 to R_2.
   c_0: goto 3 if $R_1 = R_2$ J_{123}
   c_1: $R_2 \leftarrow R_2 + 1$ S_2
   c_2: goto 0 if $R_1 = R_1$ J_{110} [Go to c_0]
   COMMENT: Now $R_1 = R_2$
   c_3: goto 7 if $R_2 = R_3$ J_{237} [Halt if $R_2 = R_3$]
   c_4: $R_3 \leftarrow R_3 + 1$ S_3
   c_5: $R_1 \leftarrow R_1 + 1$ S_1
   c_6: goto 3 if $R_1 = R_1$ J_{113} [Go to c_3]
   c_7:

2. Show that the following two problems are primitive recursive, using results from the Notes.

   a) $Bit(x, i)$ = the coefficient of 2^i in the binary notation of x. For example, the binary notation for 6 is 110, so $Bit(6, 0) = 0$, $Bit(6, 1) = Bit(6, 2) = 1$, and $Bit(6, i) = 0$ for $i > 2$.

   Solution:
   $\mathrm{Bit}(x, i) = \mathrm{rm}(\mathrm{q}(x, 2^i), 2)$, so that Bit is obtained by composition from functions which have been shown to be primitive recursive in the notes, and hence Bit is itself primitive recursive.

   b) $NumOnes(x)$ = the number of 1's in tahe binary notation for x. For example, $NumOnes(6) = 2$.

   Solution:
   $\mathrm{NumOnes}(x) = \sum_{0 \le i \le x} \mathrm{Bit}(x, i)$, so that NumOnes is obtained by a bounded sum from a primitive recursive function, so that it is itself primitive recursive.

3. Let PRIMES be the set of prime numbers.

Let $A = \{x \mid \mathrm{dom}(\{x\}_1) \subseteq \mathrm{PRIMES}\}$.
Let $B = \{x \mid \{x\}_1$ is a primitive recursive function$\}$.

Which of $A, \overline{A}, B, \overline{B}$ is recursive? Which is r.e.? Justify your answers. You may use the S-m-n Theorem (page 74) and the Kleene Normal Form Theorem (page 68). Or you may avoid using the S-m-n Theorem, and use Church's Thesis instead. (Do not use Rice's Theorem.)

Solution:
A is not r.e., and hence not recursive. $\overline{A}$ is r.e. but not recursive.

Neither B nor $\overline{B}$ is r.e., and hence neither is recursive.

To show A is not r.e. we show $\overline{H} \leq_m A$. Since H is r.e. but not recursive, it follows that $\overline{H}$ is not r.e, so this will show that A is not r.e.

Thus we want to find a total computable function f such that for all x

(*) Program $\{x\}$ fails to halt with all registers initially 0 iff $dom(\{f(x)\}_1) \subseteq$ PRIMES.

Define $g(x, y) = \{x\}_1(0) = \Phi(x, 0)$. Then g is computable, so by the Special Case of S-m-n there is a total computable $f(x)$ such that

$$\{f(x)\}_1(y) = \{x\}_1(0)$$

To establish (*), if program $\{x\}$ fails to halt with all registers 0, then $\{x\}_1(0) = \infty$, so $dom(\{f(x)\}_1 = \emptyset$, so $dom(\{f(x)\}_1) \subseteq$ PRIMES.

Conversely, if program $\{x\}$ halts with all registers 0, then $dom\{f(x)\}_1 = \mathbb{N} \subsetneq$ PRIMES.

Thus neither A nor $\overline{A}$ is recursive.

Now we show $\overline{A}$ is r.e. Let

$$R(x, y) = \exists u \leq y \neg Prime(u) \wedge T(x, u, y)$$

where T is the Kleene T-predicate. Then $R(x, y)$ is recursive (in fact primitive recursive), and

$$x \in \overline{A} \Leftrightarrow \exists y R(x, y)$$

Thus $\overline{A}$ is r.e.

To show that $\overline{B}$ is not r.e. we show $\overline{K} \leq_m \overline{B}$, which is the same as showing

$$K \leq_m B$$

Thus we want to find a total computable function $f(x)$ such that for all x

(**) Program $\{x\}$ halts on input x iff $\{f(x)\}_1$ is a primitive recursive function.

By the Special Case of S-m-n there is a total computable $f(x)$ such that for all x, y

$$\{f(x)\}_1(y) = \{x\}_1(x) = \Phi(x, x)$$

(The RHS is computable by the KNFT).

To show (**), note that if $\{x\}$ halts on input x, then $\{f(x)\}_1$ is the constant function whose value is always the number $\{x\}_1(x)$, so $\{f(x)\}_1$ is primitive recursive.

Conversely, if $\{x\}$ does not halt on input x, then $\{f(x)\}_1$ is nowhere defined, so it is not primitive recursive.

Finally we show B is not r.e. by showing

$$\overline{K} \leq_m B$$

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

(***) Program $\{x\}$ fails to halt on input x iff $\{f(x)\}_1$ is a primitive recursive function.

Again by the Special Case of S-m-n there is a total computable $f(x)$ such that for all x, y

$$\{f(x)\}_1(y) = \begin{cases} \infty & \text{if } T(x,x,y) \\ 0 & \text{otherwise} \end{cases}$$

(It is easy to see that the RHS is a computable function of x, y.) To show (***), if Program $\{x\}$ fails to halt on input x then $T(x,x,y)$ is false for all y, so $\{f(x)\}_1$ is the constant function 0, which is primitive recursive. If Program $\{x\}$ halts on input x then $T(x,x,y)$ holds for some y, so $\{f(x)\}_1(y) = \infty$, so $\{f(x)\}_1$ is not primitive recursive.

4. The class $\mathcal{E}$ of Kalmar elementary functions can be defined as follows:

   We say that f is defined from g, h, and B by *limited recursion* iff

   $f(\vec{x}, 0) = g(\vec{x})$
   $f(\vec{x}, y+1) = \min\{h(\vec{x}, y, f(\vec{x}, y)), B(\vec{x}, y)\}$

   Let $INIT(\mathcal{E}) = \{Z, S\} \cup \{I_{n,i}, 1 \leq i \leq n\} \cup \{f_+, E\}$ where $Z, S, I_{n,i}$ are the Initial Functions defined on page 57 of the Notes, and $f_+(x,y) = x + y$, and $E(x) = 2^x$.

   Definition: $f \in \mathcal{E}$ iff f can be obtained from $INIT(\mathcal{E})$ by finitely many applications of composition and limited recursion.

   Define the sequence of functions $E_0, E_1, ...$ by $E_0(x) = x$ and $E_{k+1}(x) = 2^{E_k(x)}$ for $k \geq 0$. Thus $E_k(x)$ is a superexponential function represented by a stack of k 2's with an x at the top. Note that $E_k \in \mathcal{E}$ for each k.

   For an RM program $\mathcal{P}$ let $Time_{\mathcal{P}}(\vec{x})$ be the number of steps taken by $\mathcal{P}$ before $\mathcal{P}$ halts on input $\vec{x}$, or $Time_{\mathcal{P}}(\vec{x}) = \infty$ if $\mathcal{P}$ does not halt on input $\vec{x}$.

   Definition: Let $\mathcal{C}$ be the class of all functions f such that there exists an RM program $\mathcal{P}$ and $k \in \mathbb{N}$ such that $\mathcal{P}$ computes f and $Time_{\mathcal{P}}(\vec{x}) \leq E_k(x_1 + ... + x_n)$ for all $\vec{x}$.

   Theorem: (Ritchie-Cobham) $\mathcal{E} = \mathcal{C}$.

   You are to prove the direction $\mathcal{C} \subseteq \mathcal{E}$. We start by observing that every specific function proved to be primitive recursive in the Notes *Computability Theory* is in fact in $\mathcal{E}$. This is because every application of primitive recursion can be replaced by limited recursion, since each such function $f(\vec{x}, y)$ is bounded above by $E_k(x_1 + ... + x_n + y)$ for some k, so we can take $B(\vec{x}, y) = E_k(x_1 + ... + x_n + y)$.

(a) Let $\mathcal{P}$ be an RM program, and let $STATE_{\mathcal{P}}(\vec{x}, t)$ be the number encoding the state of $\mathcal{P}$ after t steps of computation, where the initial state is $u_0 = p_0^0 p_1^{x_1} ... p_n^{x_n}$ (i.e. $x_1, ..., x_n$ are stored in registers $R_1, ..., R_n$, with program counter $K = 0$, as described on the bottom of page 58). Prove that $STATE_{\mathcal{P}} \in \mathcal{E}$. You may assume that the function $Nex(u, z)$ defined on page 59 is in $\mathcal{E}$.

Solution:
$STATE_{\mathcal{P}}(\vec{x}, t)$ can be defined as follows:

$STATE_{\mathcal{P}}(\vec{x}, 0) = p_0^0 p_1^{x_1} \dots p^{x_n}$
$STATE_{\mathcal{P}}(\vec{x}, t{+}1) = \min\{Nex(STATE)_{\mathcal{P}}(\vec{x}, t), \#\mathcal{P}, p_0^h p_1^{x_1+t+1} \dots p_n^{x_n+t+1} p_{n+1}^{t+1} \dots p_{n+m}^{t+1}\}$

where $(n+m)$ is the total number of registers used by program $\mathcal{P}$, h is the number of commands in $\mathcal{P}$, and the product of prime powers is an upper bound for $STATE_{\mathcal{P}}(\vec{x}, t+1)$. Since $Nex \in \mathcal{E}$, it follow that $STATE_{\mathcal{P}}$ is defined by limited recursion from functions in $\mathcal{E}$, and hence it is itself in $\mathcal{E}$.

b) Now use (a) to show $\mathcal{C} \subseteq \mathcal{E}$.

Solution:
Suppose $f \in \mathcal{C}$. Then there exists a program $\mathcal{P}$ and k such that $\mathcal{P}$ computes f and

$$\text{Time}_{\mathcal{P}} \le E_k(x_1 + \dots + x_n)$$

for all $\vec{x}$. Then

$$f(\vec{x}) = \big(STATE_{\mathcal{P}}(\vec{x}, E_k(x_1 + \dots + x_n))\big)_1$$

Since $STATE_{\mathcal{P}} \in \mathcal{E}$, so also $f \in \mathcal{E}$.