1. (15pts) Design a Turing machine to compute binary addition between two numbers. First, specify all assumptions about the encoding of the input and output. Then, give a high-level description of your algorithm. Finally, provide the full description of your machine, including alphabet, states, and transition function.

2. (15pts) Give a full specification of a representation scheme of Turing machines as binary strings. That is, show a procedure that transforms any Turing machine $M = (Γ, Q, δ)$ into a binary string $⌊M⌋$. It should be possible to recover $M$ from $⌊M⌋$, or at least a functionally equivalent Turing machine $\tilde{M}$.

Also, describe how your encoding fits the assumptions (1) every string represents some Turing machine, and (2) every Turing machine is represented by an infinite number of strings.

3. (10pts) Sometimes, the representation used for the input drastically changes the complexity of a problem. A number $n$ has a unary representation $[n]_{\text{unary}}$ given by $1^n$, a sequence of $n$ 1’s. Show that the problem

$$\text{UNARYFACTORING} = \{([n]_{\text{unary}}, [\ell]_{\text{unary}}[k]_{\text{unary}}) : \text{there is a number } j \in (\ell, k) \text{ dividing } n\}$$

has a polynomial-time algorithm (here the polynomial is in terms of $n + \ell + k$).

Note: The associated problem FACTORING, where the numbers are represented in binary is not currently known to have a polynomial-time algorithm (here the polynomial is in terms of $\log n + \log \ell + \log k$).

4. (15pts, required for 824 Students) A real number $r \in [0, 1]$ can be encoded in binary using an infinite number of bits: The binary sequence $x_0x_1x_2\cdots \in \{0, 1\}^\infty$ encodes the real number $\sum_{i=0}^{\infty} \frac{x_i}{2^i}$. A real number $r \in [0, 1]$ is computable if there exists a (non-halting) Turing machine which writes the binary expansion of $r$ to the output tape and for all $i$, the $i$th bit $x_i$ is written in a finite number of steps. (Note that there is no input in this case.)

Say a Turing machine computing a real number $r$ is linear if there is a constant $C \geq 1$ so that for all $i \geq 0$ the machine requires at most $C$ steps between writing $x_i$ and writing $x_{i+1}$. Hartmanis conjectured that all real numbers computable by a linear machine are either rational or transcendental.

Define a linear Turing machine to compute the transcendental number (in binary)

$$0.10110111011110\cdots 0 \underbrace{11\cdots 1}_k 0 \underbrace{11\cdots 11}_k 0 \underbrace{11\cdots 111}_k 0 \ldots$$

HINT: There is a machine that outputs one bit per time step.