Amin Zolfagharian - 610300118

Problem 1.

By characterisation of r.e. sets we know that if A is r.e. then A is the range of a unary total computable function. Let that function be f. Thus we have:

$$A = \{f(0), f(1), f(2), \dots\}$$

But f might have repetitions. Now we define total computable g such that A is the range of g, and g is injective.

$$g(x) = \mu_z(\forall_{x<z}(f(x) \neq f(z)))$$

Since f is computable and both μ and $\forall$ are computable, then g is also computable. Also g is total since A is infinite and range of f is A hence there will always exists such z, that $\forall_{x<z}(f(x) \neq f(z))$. Also $Ran(g) = A$, since for any $a \in A$, there exists i such that $f(i) = a$. Let j be such that $f(j) = a$ and for any $f(i) = a$, we have $j < i$. Then we have $g(j) = a$. This completes the proof as g is a total unary function that enumerates A without repetitions.

Problem 2.

Assume A is decidable. We define the function g as below:

$$g(x) = \begin{cases} x & x \notin A \\ x+1 & x \in A \end{cases}$$

Since A is decidable, then g is also decidable. Therefore there exists some $e \in \mathbb{N}$ such that $g \cong \phi_e$.

$$\phi_e(e) = e \iff g(e) = e \iff e \notin A \iff \phi_e(e) \neq e$$

Which is a contradiction. This shows that A cannot be a decidable set.

Problem 3.

Let P be a program with n instructions. We can create another program Q as below. Let r be a register that is not used in P, and let $P = I_1 I_2 \dots I_n$. Let:

$$I_1, S(r), I_2, S(r), \dots I_n, S(r), T(r, 1)$$

Now Q runs P and outputs the length of computation. Now we will introduce a program that can decide whether $P(0)$ halts or no, which is an undecidable problem. consider $P(0)$, and make Q as described, then by computablity of busy beaver function, we know that if a program with at most $2n+1$ instructions halts, then it would have an output equal or less than $B(2n+1)$. Now we by construction we know that if P halts, then Q halts, and vice versa, and since $Q(0) \leq B(2n+1)$, we know that the length of computation for P is at most $B(2n+1)$. Therefore we can run $P(0)$ for $B(2n+1)$ steps, if it halts, then we answer yes, and if it doesn't halt, then it will never halt, since for any program with at most n instructions, there is a corresponding program that computes its length of computation, and is calculated in $B(2n+1)$. Now since we just proved that halting problem is decidable, we arrive at a contradiction, which is due to assumption of computablity of B, hence $B(n)$ is not computable.

Problem 4.

(i) Let A be the set of all URM-programs that have at most n instructions. For any instruction we have 4 choices, therefore all of these programs are a combinations of at most n of these 4 instructions. We have at most 5^n combination of these instructions (each instruction can be a J or S or T or Z or empty, it is just an upper bound). Each instruction can refrence 3 registers at most, Which means in any program at most $3n$ registers are used. Since it doesn't matter which set of $3n$ registers we use from the infinite registers that we have, we only need to consider the combinations of these in instructions. Which is at most $3n^{3n}$. This shows that we have at most $5^n \cdot 3n^{3n}$ programs with at most n instructions. Which shows that A is a finite set.

(ii) Let $B(n) = y$ for some $y \in \mathbb{N}$. Then there exists some program P such that $P(0) \downarrow y$, and P has at most n instructions. Now consider A be the programs with at most $n+1$ instructions. therefore $P \in A$. And since $B(n+1)$ is the biggest output from programs in A therefore $B(n+1) \geq y = B(n)$.

(iii) For $n = 1$ consider the program $S(1)\ S(1)\ \ldots\ S(1)$. Which outputs 6. For $n > 1$ consider the program below:

```
Start: S(2)
       S(2)
Loop:  J(2, 3, End)
       S(3)
       S(1) (n times)
       J(1, 1, Loop)
End
```

This program has exactly $n+5$ instructions, and it compeltes the loop twice, and in each loop it adds n to register r_1. Therefore it generates $2n$. Which proves that $B(n+5) \geq 2n$.

(iv) We define g in a recursive manner:

$$g(0) = f(0) + 1$$
$$g(n+1) = h(n, g(n)) = g(n) + f(n+1) + 1$$

Since f is computable and total, then g is also total and computable and increasing. And it is also easy to see that for any $n \in \mathbb{N}$, we have $g(n) > f(n)$.

(v) Let f be a computable function. Sicne there exists some computable g such that g dominates f. Now let P be the program that computes g. And it has n instructions. This means that $g(0) \leq B(n)$. Now some $k > n + 5$.

$$f(2k) < g(2k)$$

We can compute $g(2k)$ with a program Q with $k+5+n$ instructions. We use part (iii) to first create $2k$ in the first register. Then we use the program P. Q is a program with $k + 5 + n$ instructions and since $k > n + 5$, then Q has at most $2k$ instructions, therefore we have:

$$f(2k) < g(2k) \leq B(2k)$$

Now suppose we know that $g(m) \leq B(m)$, since there is some program with Q that first creates m in the first register and then operates on it with P. If we add a $S(1)$ to the start of Q then we have another program with at most $m+1$ instructions, that computes $g(m+1)$, therefore $g(m+1) \leq B(m+1)$. This shows that for any $m > 2k$ we have:

$$f(m) < g(m) \leq B(m)$$

Thus B dominates f. Therefore B dominates every computable function, including itself. Let $f = B$:

$$\exists n_0; \forall n > n_0 : B(n) < B(n)$$

This contradiction shows that B cannot be a computable function.