CS533: Homework 9

When you prove a language is undecidable using a proof by contradiction, make sure you clearly say what you are assuming, clearly give your construction, and say why there is a contradiction (explain what the construction is doing), and what assumption is thus false.

You can make use of any of the languages that are proved undecidable in Chapters 4 and 5, but cite which theorem it is from.

Question 1: Exercise 5.1

Hint: find a language from Chapter 5 that is undecidable that is about CFG and make a reduction from it to $EQ_{CFG}$.

Question 2: Exercise 5.2

You need to give a construction of a TM. First, explain what this TM needs to check. Second, give a construction of your TM. Give a high level description, like the descriptions in this chapter. Clearly say what the input is. Argue why your TM is correct. Note that you do not need any of the tools in this chapter to solve this question.

Question 3: Exercise 5.3

Question 4: Exercise 5.9

Construct a reduction from $A_{TM}$. Make sure you argue why your TM decides $A_{TM}$. Your proof will have a similar structure as the proof of Theorem 5.3, in which you give the construction of a TM, which itself constructs a TM.

Question 5: Problem 5.14

I found this one to be relatively straight-forward.

Question 6: Exercise 5.19

Give an algorithm (it will be REALLY simple). But, make sure you justify why the algorithm is correct.

Question 7: Exercise 5.17

I did this by considering 3 cases about whether certain tiles exist.

Question 8: TM

Easy question: Consider the TM in the figure, in which the c is the accept state and the input alphabet is $\{1, 2\}$ and ‘_’ is the empty space character that follows the input. What language does it recognize? Is this TM deterministic? Is the TM a decider or just a recognizer? What class does this language fall into?
Question 9: Computation History

Easy question: The string 121 is in the language of the TM from the question above. Give an accepting computation history for the TM on input 121. For the states, simply refer to them as \( a \), \( b \) and \( c \), rather than \( q_a \), \( q_b \), and \( q_c \). When the head of the TM is pointing to the tape location past the end of the input, make sure to include a blank character.

Question 10: First approximation of PCP

For the TM above, capture how it computes as a problem in PCP, in which tiles capture how one configuration can change into another.

In this first approximation, use the top and bottom of tiles to express entire individual configurations. So, a sequence of tiles can represent a computation history.

Here we will be ignoring that we need an infinite number of tiles. Just include the ones needed for the computation history of the previous question. Hint: the number of tiles needed is the length of the accepting computation history plus 1.

Also, show how your tiles are a solution to the PCP problem.

Note that one of your tiles will be consistent with Part 1 of the construction for Theorem 5.15 (that PCP is undecidable).

Question 11: Towards Full MPCP

Let’s fix up the problem with the previous question, where you need an infinite number of tiles to express how any possible configuration can change to another one (via the transition function). Since the configuration includes the tape contents, and you have to account for anything that might be on the tape, you need an infinite number of tiles.

Let’s ignore the problem with the end tile. Give tiles to fully capture the transitions in the delta function (even ones not needed for this example). What other tiles do you need?

Note that this is Part 1 through 4 of Theorem 5.15.

Show how these tiles can be used to find an accepting computation. You only need to show the sequence of tiles to cover the first 3 configurations in the bottom string.