COMP 314 Homework Assignment C

Chapter 5

Question C1. (Ungraded) As suggested on page 95, use JFLAP to implement a Turing machine `swapCandG`, that changes every "`C`" to a "`G`", and every "`G`" to a "`C`". Test your machine on multiple inputs.

Question C2. (Ungraded) Using JFLAP, create your own version of the `binaryIncrementer` machine described on page 97. Construct your machine without looking at the solution given in Figure 5.10.

Question C3. (15 points) Use JFLAP to create a `binaryDecrementer` machine, whose behavior is similar to the incrementer in the previous question except that it to decrements binary numbers rather than incrementing them. Your machine's output is permitted to contain leading zeros. For example, "`x10x`" becomes "`x01x`".

Question C4. (Ungraded) Suppose we wish to construct a Turing machine `noMoreCsThanGs`, which accepts genetic strings whose number of `C`s is no more than the number of `G`s. This can be achieved by making some simple changes to the `moreCsThanGs` machine of Figure 5.9. Describe the changes that would be required. (There is no need to draw a diagram of the resulting machine.)

Question C5. (5 points) The proof on page 107 was written for the specific case of a machine M with two tapes and 129 symbols in its alphabet. The resulting simulator, M', used a larger alphabet. In general, how many symbols would be in the alphabet of M' when the same kind of simulation is used for an M with k tapes and an alphabet of s symbols?

Question C6. (Ungraded) Let M be a 6-tape, single-head Turing machine that employs an alphabet of 10 symbols. Suppose we use a technique similar to the proof on page 107 to show that M can be simulated by a 2-tape, single-head Turing machine M'. How many symbols are in the alphabet of M'? Explain your answer.

Note: M' has *two* tapes, which differs from the proof on page 107. The intention of this question is that the simulation should use both of these tapes, approximately equally. You could, of course, use exactly the same simulation as the proof on page 107, by employing only one of the tapes on M'. But that would be no fun.

Question C7. (5 points) The proof on page 110 was written for the specific case of a machine M with two tapes and two independent heads. Suppose

instead that M has k tapes, k independent heads, and s symbols in its alphabet. How many tapes and symbols are required by the corresponding single-head simulator M'?

Question C8. (Ungraded) Suppose M is a standard (single-tape) Turing machine using the five-symbol alphabet $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{e}\}$. Give a high-level description of how to simulate M using a standard (single-tape) Turing machine M' that employs the four-symbol alphabet $\Sigma' = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$. Your simulation need only work correctly for input and output strings that contain no "`e`" characters. More precisely, M' must have the property that if $I \in \Sigma'^*$ and $M(I) \in \Sigma'^*$, then $M'(I) = M(I)$.

Question C9. (20 points) On page 115, it was stated that any CPU instruction could be implemented with a Turing machine building block. This question asks you to create such building blocks for two common CPU instructions.

(a) Construct a Turing machine that implements the SHIFT instruction. The input is a binary string flanked by `x` characters, and the output is the binary string shifted right by one bit. The right-most bit is deleted, the string is padded on the left with a `0`, and the final result is flanked by `x` characters. Example: "`x11001x`" becomes "`x01100x`".

(b) Construct a Turing machine that implements the logical AND instruction. The input is two binary strings separated and flanked by `x` characters. The output is "`x1x`" if both binary strings were nonzero, and "`x0x`" otherwise. Examples: "`x101x0100x`" becomes "`x1x`"; "`x00x010x`" becomes "`x0x`".

Question C10. (Ungraded) Write your own version of `simulateTM.py` (see Figure 5.19), without consulting the version provided with the book materials.

Question C11. (15 points) Consider the `binaryDecrementer` Turing machine you constructed for an earlier question. Give an ASCII description desc(`binaryDecrementer`) of this machine, using the construction suggested by Figure 5.18.

Question C12. (Ungraded) Conduct experiments with `simulateTM.py`, as suggested on page 117. In particular, try out

```
>>> simulateTM(rf('containsGAGA.tm'), 'TTGAGATT')
```

and

```
>>> simulateTM(rf('binaryIncrementer.tm'), 'x101x')
```

Question C13. (Ungraded) A quantum algorithm is an algorithm (i.e. a computer program) that runs on a quantum computer. Is it likely that someone will invent a quantum algorithm that solves the problem YesOnString? Why or why not?

Question C14. (Ungraded) On page 15, Chapter 1 described an intractable problem known as *multiple sequence alignment*, which we will now abbreviate as MultSeqAlign. Suppose we augment a standard modern computer with a black box that can instantaneously solve any instance of MultSeqAlign, and equip the computer with as much memory as necessary. Is this type of augmented computer Turing-equivalent to the class of multi-tape Turing machines? Explain your answer.

Chapter 6

Question C15. (Ungraded) Conduct all the experiments suggested in Chapter 6, including:

(a) Experiment with using `exec()` on various snippets of Python.

(b) Use `universal.py` to determine the output of `countLines.py` on several different inputs.

(c) Experiment with simulating altered versions of programs by trying various inputs to `simAltKangarooToKoala.py`.

Question C16. (5 points) Let U be a universal Turing machine. Suppose we wish to use U to simulate the operation of the `binaryIncrementer` machine (page 97) on input "`x1001x`". Describe the input string I that should be given to U in order to achieve this simulation.

Question C17. (10 points) Write a Python program `repeat.py` that takes as input a single string parameter S. The parameter S encodes two strings P and I in the usual way: $S = \text{ESS}(P, I)$. The output of `repeat.py` is the concatenation of $P(I)$ with itself.

Question C18. (Ungraded) Write a Python program `applyBothTwice.py` that takes as input a single string parameter S. The parameter S encodes three strings P, Q, I in the following way: $S = \text{ESS}(\text{ESS}(P, Q), I)$. The output of `applyBothTwice.py` is $Q(P(Q(P(I))))$.

Total points on this assignment: 75