Rhea Mae Edwards

CS 321: Homework #7

For reference, here is the *CFL pumping lemma game* (for language A):

1. Adversary picks a number $p \geq 0$.
2. You pick a string $s \in A$, such that $|s| \geq p$.
3. Adversary breaks s into $s = uvwxy$, such that $|vwx| \leq p$ and $|vx| > 0$.
4. You pick a number $i \geq 0$. If $uv^iwx^iy \notin A$, then you win.

If you can describe a strategy in which you always win, then A is not context-free.

1. Show following are not context-free

a. $\{xcy \mid x, y \in \{0, 1\}^*$ and $\text{bin}(x) + 1 = \text{bin}(y)\}$

Hint: Note the similarity between this and $\{xcx \mid x \in \{0, 1\}^*\}$.
How would you proceed in that case?

1. Adversary picks a number $p \geq 0$.

2. I pick $s = x^pcy^p = 0^p1^p{+}1c0^p1^p$ where $s \in A$ and $|s| \geq p$.

3. Adversary breaks $s = uvwxy$, such that $|vwx| \leq p$ and $|vx| > 0$.

4 I pick $i = 2$.
$uv^iwx^iy = uvvwxxy$
$|vwx| \leq p$, so y would have an extra unwanted 0 at the beginning of that part of the string or x with an extra 1 at the end of that part of the string, which would make the string not in the language. Meaning that this language is not context-free.

b. $\{a^ib^jc^k \mid i < j < k\}$

Hint: Your strategy in step 4 can (should) depend on the adversary's move in step 3.

1. Adversary picks a number $p \geq 0$.

2. I pick $s = a^p\, b^pc^p$ where $s \in A$ and $|s| \geq p$.

3. Adversary breaks $s = uvwxy$, such that $|vwx| \leq p$ and $|vx| > 0$.

4. I pick $i = 2$.
$uv^iwx^iy = uvvwxxy$
Since $|vwx| \leq p$, this string cannot hold any a's or c's.
There would be a greater amount of a's and/or b's than c's of the string, which will make this string not in the language. Meaning that this language is not context-free.

2. Turing Machine deciding $\{a^n b^m c^{nm} \mid n, m \in \mathbb{N}\}$

Note: Possibly can do this in about 9 states.

- Give a plain-language description of how the TM head moves and modifies the tape contents.
- Give a complete transition table.
  (You can assume the TM has a way of detecting the left end of the tape)
- Give your states human-readable names, so that the correspondence between your plain-language description and transition table is clear.

<u>Plain-Language Description:</u>

A way that this Turing machine would move its head and modify its tape contents is explained by the following…

- First the Turing machine will read the first character of the tape.
  - If the character is an *a*, then the Turning machine will replace that *a* with an *x*, and move ahead and read it's way through until it finds a *c*.
    - If it reads another *a*, it will just pass right through it.
    - If it reads a *b*, it will pass right through it.
    - If it reads a *c*, then it will replace that *c* with an *x*, and then move its way to the beginning of the string (left end of the tape).
      - It will then pass through any *x*'s that it sees, and when it reads another *a*, it will repeat this whole first process over and over again, until it reads a *b*.
        - If there is a *c* at this part, then the Turing machine will reject the string, because there cannot be any *c*'s before reading a *b* and/or you cannot have any *c*'s of a string that is part of this language if there are not any *b*'s present in the string overall.
    - If it reads a blank character, then the Turing machine will accept the string, because the machine will only contain that *a*, which is a valid string in the language.
  - If the character is a *b*, then the Turning machine will just read through the string till it reaches a blank space/the end of the string.
    - If it reads another *b*, it will keep on moving through the string, making sure that the rest of the characters are *b*'s, and if so, the Turing machine will accept the string because that would be a part of the language.
    - If it reads a blank space, the string will be accepted since a single *b* is a valid string of the described language.
    - If it reads an *a*, then such a string will be rejected by the Turing machine, since such a string is not a part of the language, because you cannot have an *a* after a *b* character of a string in this language.
    - If it reads a *c*, then such a string will be rejected by the Turing machine, because you need at least an *a* at the beginning of the string in order to have any *c*'s at the end of a string in this language.
  - If the character is a *c*, then such a string will be rejected by the Turing machine, because there needs to be at least an *a* and a *b* of a string in this language in order to any *c*'s of a string in this language, and you also cannot start of string in this language with a *c* either.
  - If the first thing that the Turing machine reads is a blank space, then the Turing machine will accept the string, because an empty string is a valid string in this language.

- Next, after all the *a*'s of the beginning have been replaced with *x*'s, and all of the corresponding *c*'s also have been replaced with *x*'s, then a *b* is read, it will consider the following…
  - Once the Turing machine reads a *b*, it will replace it will an *x*, then move forward/right to the next *x* (that was once a *c*), and mark it with a symbol (such as ~), then move forward/right till it reaches another *x*, then mark it with the same symbol till it reaches another type of character or a blank space along the tape.
    - If it reaches a *c*, it will go back/right along the tape, till it reaches another character *b*.
      - If it doesn't see another *b*, then the Turing machine will reject the string, because there can't be more *c*'s if there are no more *b*'s in this process, because then the multiplication does not work that is described in the language; there would then be more *c*'s than the multiplication of the number of *a*'s and *b*'s it would have called for.
  - After reading a *b* it will go forward/right along the string and find the first marked *x*'s and then for each marked *x*, re-mark/change it's mark, and then it will go forward/right along the string and find the next *c* and replace it with an *x* as explained previously before.
    - All conditions apply as above.
  - This process is then repeated till all the Turing machine reads are marked and unmarked *x*'s, and it so, the Turing machine will accept the string, because then it will be in the language.

<u>Transition Table:</u>

If in state…	**Reading…**	**Do…**
"Beginning of the tape"	a	
	b	
	c	
	□	
"Looking for *a*"	a	
	b	
	c	
	x (unmarked), x (marked)	
	□	
"Looking for *b*"	b	
	x (unmarked), x (marked)	
	a, c, □	
"Looking for *c*"	c	
	a, b, □	
	x (unmarked), x (marked)	
"Looking for *x* (unmarked)"	x (unmarked), x (marked)	
	a, b, c, □	
"Looking for *x* (marked)"	x (unmarked), x (marked)	
	a, b, c, □	
"Checking for all *x* (unmarked and marked) string"	x (unmarked), x (marked)	
	a, b, c, □	

Didn't have time to completely finish the transition table…

Fully explained in the "Plain-Language Description" section though.