CS 581: Theory of Computation
James Hook
Assignment 9
Due at the final class meeting.

1. Argue that your favorite programming language is an Acceptable Programming System.

2. In Kozen's text *Theory of Computation* he gives applications of the Recursion Theorem as Lecture 34. In that Lecture he gives an example of a self-printing program. Such programs are called "quines". I have adapted Kozen's quine for C to work on my computer as follows:

```
#include<stdio.h>
char *s="%cinclude<stdio.h>%cchar *s=%c%s%c;%cmain(){printf(s,35,10,34,s,34,10,10);}%c";
main(){printf(s,35,10,34,s,34,10,10);}
```

Note that the decimal numbers 10, 34, and 35 represent the ASCII characters newline, double quote, and sharp sign.

Write a "quine" (a self-printing program) for another programming language.

3. This problem follows Kozen's formation, but it is not necessary to read Kozen's text to solve the problem.

Kozen illustrates the application of the recursion theorem by considering the following program transformation, which takes a program P and returns a new program that contains P:

$$P \mapsto \lambda x. \begin{cases} 1, & \text{if } x = 0 \\ x \cdot P(x-1), & \text{otherwise.} \end{cases}$$

(Note: In this example λ is being used as a generic notation for anonymous functions. This is not an official λ-calculus program.)

Adapt this fixed-point construction to your favorite programming language. Use this construction and the fixed-point version of the recursion theorem presented in class to justify that the factorial function is definable in your favorite programming language. (This should be the same language you previously showed was an acceptable programming system.)

Pay careful attention to when you are generating and/or transforming a program and when you are executing the program.

Note that your solution will not directly define the factorial function, but it will prove that, as a consequence of the recursion theorem, the factorial function can be defined in your favorite programming language.