Report for the IBM 650 Emulator in Python
Topic Paper #9

Put your name here
CS-3210-01
2/12/09
Survey of Programming Languages
Spring 2009
Computer Science, (319) 360-8771
alexdlaird@cedarville.edu

Grading Rubric		Max	Earn
	On Time/Format	1	
	Correct	5	
	Clear	2	
	Concise	2	
	Total	10 pts	

Peer Reviewer

ABSTRACT
This paper discusses the completed IBM 650 emulator developed using the Python programming language.

Keywords
IBM 650, Python, Emulator

1. OVERVIEW
The emulator has the capabilities of reading and writing from and to a text file (if two arguments are specified on instantiation), reading from a text file and writing to the command line (if only one argument is given), or reading and writting from and to the command line (if no arguments are given).

The first batch of messages, up until a "+9999999999" is received and stored in a list that emulates the memory. The second batch of messages, up until the stop command "+9000000000" followed by the program terminating "+9999999999" is stored in a list that handles the messages. The connection to the input stream is left open for continued reading if more read commands are given.

2. PARSING
Each message is parsed according to the IBM 650 specifications given for this project. The lead ~# (where ~ is + or – and # is some number) is the command operator; the first set of ### is the memory location of the first value, the second set of ### is the memory location of the second value, and the final set of ### is the destination memory location.

3. TESTING
The biggest problems run into was in the test programs that were given to run. Certain test programs outputted the destination when they should have outputted the first memory location of the message. As this is the case, some of the output may be incorrect, though I tried to accommodate for this error as much as I could. For the most part, I followed the convection of the "exercise" test files provided and not the format used in many of the student test files.

4. ERROR HANDLING
As the specifications told us that the only input/output the program should do should be strictly the input of the program and the direct output (that being the answer) of the program, errors are handled silently. If an error is received, the program will terminate gracefully without causing the shell (or your memory) to explode. However, if you are running the application from the Python shell, certain exit codes will be given in the Traceback that can be used to identify what error exactly went wrong. These error codes are defined at the top of the code for the emulator. So errors *are* caught, but the method of handling is to simply ignore and terminate the program.

5. MOST CONCLUSIVE TEST
The overall best test that was run was created by Ryan Morehart of our class. The program conclusively tested every single operation the emulator was possible of. The output for this test file should is probably the most reputable and is also the most accurate. It has been provided in my main folder as "fib_out.txt," and Morehart's test program is "fib.txt."

Alex Laird, Cedarville University, Cedarville, Ohio, 45314