MP 7 – C vs. Python

CS 421 – Spring 2011

Revision 1.0

Assigned March 10, 2011
Due March 16, 2011, 23:59
Extension 24 hours (20% penalty)
Total Points 20

1 Change Log

1.1 Fixed a bug in C code (isintstring on empty string), added a function to read a line from stdin.

1.0 Initial Release.

2 Objectives and Background

The purpose of this MP is to have you learn a little bit of Python, and compare it to C. You will write the same program in both languages.

The problem is simple enough that you can learn enough Python on your own to solve it (although we will provide some hints, as well as some code, for both the Python and C solutions). We expect that you will find that Python makes some things a lot simpler; our solutions are about 100 lines of C and 40 lines of Python.

3 Collaboration

Collaboration is NOT allowed in this assignment.

4 What to submit

You will submit your Python and C files using the `handin` program as usual. The files you submit are called `mp7.py` and `mp7.c`. Rename `mp7-skeleton.py` to `mp7.py`, `mp7-skeleton.c` to `mp7.c` and start working from there.

5 Instructions

EWS machines currently have only an old version of Python installed (Python 2.4.3). Python bytecode is not portable across different Python versions, so in order to test your solution agains ours **you must either use an EWS machine** or install Python 2.4 on your own machine.

6 Problem

The problem is to write the following program in both C and Python:

Your input is a sequence of lines each containing either an integer (decimal digits), a floating-point number (decimal digits with decimal point), or a string (anything else). You may assume no line contains more than 99 characters, and there are no more than 1000 lines. First read these lines into an array; call the array V, and its length n. Iterate over V and do the following for each entry: If that entry is not an integer, do nothing. If it is an integer, say i, and if i is in the range of subscripts of the list (i.e. $0 \leq i < n$), then perform the following action *on* $V[i]$: If it is an int, add 1; if it is a float, double it; if it is a string, concatenate it with itself. Then print V, one entry per line.

For example, this input file:

```
1
two
3
4.0
fivesix
7.0
1
nine
8
```

should produce this output:

```
1
twotwotwotwo
3
8.0
fivesix
7.0
1
nine
9
```

Note that index 1 occurred twice in the input, so the entry at position 1 was modified twice. Note also that the final input value was 8, which was also at index 8, so it "modified itself."

1. (10 pt) Write the above program in C. There are certain constraints on how you should write it. You should store the input values in an array of structs, where each struct contains a "tag" — a small integer indicating what type of value is in the struct — and a union type containing the value (int, float, or char pointer). Although you could statically allocate space for strings (since there can be at most 100×1000 characters), we do not allow that: you can have a 100–character input buffer, but stored strings must be allocated in the heap (using `malloc`).

   This program should be defined in a file `mp7.c`. Your program will get its input from standard input; you can use the provided function `readline(buffer)`, where `buffer` is a 100–character array, to read each line; `readline` returns `EOF` when it reaches end of file.

   We also provide functions `isintstring` and `isdoublestring` that test whether a string is formatted as an integer or floating-point number; you can use library functions `atoi` and `atof` to convert strings to ints or doubles, respectively.

   We offer two more hints: You will probably want to use functions `strcat` and `strcpy`, and possibly `strlen`. When allocating space for a string `s` using `malloc`, remember that you have to allocate `strlen(s)+1` bytes, the extra one being for the terminating null character.

2. (10 pt) Write the above program in Python. You will need to learn enough Python to do this. The essential Python tutorial is at `http://docs.python.org/tutorial/`, but it's kind of long; the one at `http://www.tutorialspoint.com/python/` is good — just ignore the first few pages.

   This program should be defined in a file `mp7.py`. Specifically, that file should contain a function `run` with no arguments. Program should read its input from standard input and write result to the standard output.

   Again, we expect you to solve this in a particular way. We provide an input routine `readlines` to read standard input to a list of strings. You should then convert each string to a value; since Python is dynamically typed, you can just store the value back into the list. We also provide functions `isintstring` and `isfloatstring`, analogous to the functions of these names we provided in C; use Python functions int(s) and float(s) to convert a string s in the proper format to the corresponding type of value. You should calculate the values into a list, then use `print` on each element of the list to print it.

   Here are a few more hints:

   - The `range` function returns a list of integers, and the statement `for i in range(...):` iterates over the values in that range.
   - Lists are mutable, so you can reassign their values. So within the for loop just mentioned, you can write statements `input[i] = ...`.
   - The function `type` gives the type of any value. It returns it as a special type of object, but you don't have to know what that looks like; if you want to test whether, for example, `x` is an int, just write `if type(x) == type(0).` (Do not confuse this test with the function `isintstring`. The latter takes a string argument and tells you whether that string contains only decimal digits. The test just shown takes any value `x` and says whether it is an integer. `type("342")` is string, not integer, but `isintstring("342")` is `True`.)

7 Testing your code

The build process remains the same. Run `make` to compile your code and `grader` to test it.

To add new tests, simply create a file with `.in` extension in `mp7grader/tests` directory.

To run your Python program, execute this command:

```
$ python mp7.py <inputfilename
```

To run your C program, execute this command:

```
$ make && ./student <inputfilename
```

For example:

```
$ make && ./student <tests/test1.in
cp mp7.c student.c
cp mp7.py student.py
gcc --std=c99 -o student student.c
1
twotwotwotwo
3
8.000000
fivesix
7.000000
1
nine
9
```

```
$ python mp7.py <tests/test1.in
1
twotwotwotwo
3
8.0
fivesix
7.0
1
nine
9
```