CS 112 – Debugging Lab

Debugging is a methodical process of finding and reducing the number of bugs, or defects, in a computer program. Debugging is, in general, a cumbersome and tiring task. The debugging skill of the programmer is probably the biggest factor in the ability to debug a problem, but the difficulty of software debugging varies greatly with the programming language used and the available tools, such as debuggers. Debuggers are software tools which enable the programmer to monitor the execution of a program, stop it, re-start it, run it in slow motion, change values in memory and even, in some cases, go back in time.

An experienced programmer often knows where errors are more likely to occur, based on the complexity of sections of the program as well as possible data corruption. For example, any data obtained from a user should be treated suspiciously. Great care should be taken to verify that the format and content of the data are correct. Data obtained from transmissions should be checked to make sure the entire message (data) was received. Complex data that must be parsed and/or processed may contain unexpected combinations of values that were not anticipated, and not handled correctly. By inserting checks for likely error symptoms, the program can detect when data has been corrupted or not handled correctly.

If an error is severe enough to cause the program to terminate abnormally, the existence of a bug becomes obvious. If the program detects a less serious problem, the bug can be recognized, provided error and/or log messages are monitored. However, if the error is minor and only causes the wrong results, it becomes much more difficult to detect that a bug exists; this is especially true if it is difficult or impossible to verify the results of the program. This step is often the most difficult (and therefore rewarding) step in debugging. The idea is to identify what portion of the system is causing the error. Unfortunately, the source of the problem isn't always the same as the source of the symptoms. For example, if an input record is corrupted, an error may not occur until the program is processing a different record, or performing some action based on the erroneous information, which could happen long after the record was read.

This process often involves iterative testing. The programmer might first verify that the input is correct, next if it was read correctly, processed correctly, etc. For modular systems, this step can be a little easier by checking the validity of data passed across interfaces between different modules. If the input was correct, but the output was not, then the source of the error is within the module. By iteratively testing inputs and outputs, the debugger can identify within a few lines of code where the error is occurring.

Skilled debuggers are often able to hypothesize where the problem might be (based on analogies to previous similar situations), and test the inputs and outputs of the suspected areas of the program. This form of debugging is an instance of the scientific method. Less skilled debuggers often step sequentially through the program, looking for a place where the behavior of the program is different from that expected. Note that this is still a form of scientific method as the programmer must decide what variables to examine when looking for unusual behavior. Another approach is to use a "binary search" type of isolation process. By testing sections near the middle of the data / processing flow, the programmer can determine if the error happens during earlier or later sections of the program. If no data problems are detected, then the error is probably later in the process.

(source: en.wikipedia.org/wiki/Debugging)

There are two basic ways to perform debugging:

- using print statements to monitor the state of variables during the operation of the system.

- using debugging systems (e.g., the command line debugging facility (pdb) and the debugging capability built in to IDLE).

The assignment

The first method of introducing print statements is a basic way that you probably have already done throughout the class. Using print statements will help you understand the state of variables to validate your assumptions.

This lab involves an introduction to the second method of using debugging systems. Debugging systems are present in all modern languages and provide a good way to watch your code, understand what is happening and find logical errors. The following two links provide resources for the two debugging tools mentioned in the second bullet above:

- PDB – http://www.ferg.org/papers/debugging_in_python.html
- IDLE DEBUGGER – http://www.python.org/idle/doc/idle2.html#Debugger

Following is a file that contains a logic error. Study the code and output carefully to determine the problem, then demonstrate the use of either PDB or the IDLE debugger (or both) in locating the logic error.

debugMe.py (http://cs.gmu.edu/~dfleck/classes/cs112/spring08/labs/debugger/debugMe.py)

```
# DebugMe.py - This file contains code with a logic error.
# The general idea of this program is to convert from one currency
to another


# Return the amount
def calc_dollar_amt(cda_rate, foreign_amt):
    cda_dollar_amt = foreign_amt / cda_rate
    return cda_dollar_amt


def main():
    currency = raw_input("Enter the currency type: ")
    for i in range(3):
        prompt = " Enter exchange rate, %s amount : " %(currency)
        rate, amt = input(prompt)
        dollar_amt = calc_dollar_amt(rate, amt)
        print "%0.2f %s = $%0.2f" % (amt,currency,dollar_amt)

main()
```

output:

```
Enter the currency type: Yen
 Enter exchange rate, Yen amount : 5.5, 20
20.00 Yen = $3.64
 Enter exchange rate, Yen amount : 6, 20
20.00 Yen = $3.00
 Enter exchange rate, Yen amount : 6, 22
22.00 Yen = $3.00
>>>
```

What to turn in:

- Screen shots showing the cause of error displayed using PDB or IDLE's debugger.
- Additionally add a text note into your submission explaining how your screen shots show the cause of the error. (Specifically, I do not want to see a screen shot showing the output of the program (I already provided that to you) – I want to see the CAUSE of the error.