Functional debugging involves the verification of input/output parameters. Functional debugging is a static process where inputs are supplied, the system is run, and the outputs are compared against the expected results. Four methods of functional debugging are presented in this section, and two more functional debugging methods are presented in the next chapter after indexed addressing mode is presented. There are two important aspects of debugging: control and observability. The first step of debugging is to **stabilize** the system. In the debugging context, we stabilize the problem by creating a test routine that fixes (or stabilizes) all the inputs. In this way, we can reproduce the exact inputs over and over again. Stabilization is an effective approach to debugging because we can control exactly what software is being executed. Once stabilized, if we modify the program, we are sure that the change in our outputs is a function of the modification we made in our software and not due to a change in the input parameters. When a system has a small number of possible inputs (e.g., less than a million), it makes sense to test them all. When the number of possible inputs is large we need to choose a set of inputs. There are many ways to make this choice. We can select values:

- Near the extremes and in the middle
- Most typical of how our clients will properly use the system
- Most typical of how our clients will improperly attempt to use the system
- That differ by one
- You know your system will find difficult
- Using a random number generator

To stabilize the system we define a fixed set of inputs to test, run the system on these inputs, and record the outputs. Debugging is a process of finding patterns in the differences between recorded behavior and expected results. The advantage of modular programming is that we can perform modular debugging. We make a list of modules that might be causing the bug. We can then create new test routines to stabilize these modules and debug them one at a time. Unfortunately, sometimes all the modules seem to work, but the combination of modules does not. In this case we study the interfaces between the modules, looking for intended and unintended (e.g., unfriendly code) interactions.

EdX is a non-profit created by founding partners Harvard and MIT whose mission is to bring the best of higher education to students of all ages anywhere in the world, wherever there is Internet access. EdX's free online MOOCs are interactive and subjects include computer science, public health, and artificial intelligence.

(http://www.facebook.com/EdxOnline)

(https://twitter.com/edXOnline)

(https://plus.google.com/108235383044095082735/posts)

(http://youtube.com/user/edxonline)

Help