11.1

The printf will never be executed, because “i” is initialized to 100 and the condition in “for” loop will check fail at first time.

If we modify the condition to “i >= 0”, then the “for” loop will be an infinite loop because “i” is an unsigned int which can’t not be negative.

To correct the code to print all numbers from 100 to 1, we can change the condition to “i > 0”.As Follows:

1 unsigned int i;

2 for (i = 100; i > 0; --i)

3 printf(“%d\n”, i);

11.2

The bugs largely depend on the type of application which is being diagnosed. There are two main reasons as follows.

1. Random variable: The program goes into different branches according to user input or random variables that is generated in the program, then the bugs will appear in different places.

2. Memory leak: The program may have run out of memory. The crash time depends on the numbers of processes running at that particular time. This also include heap overflow or stack overflow.

It’s also caused by the other programs or external module that this program depends on. For example, if our program depends on some system attributes which may be modified by other programs, then this interference will lead to crash.

11.3

There are two types of testing we should do:

1. Input/Output validation:

We should validate both input and output to make sure that each are valid as follows:

(1)Check whether input is with the board limit:

Attempt to pass in negative numbers.

Attempt to pass in numbers which are larger than the width of the board.

Attempt to pass in numbers which are larger than the height of the board.

Depending on the implementation, the game program should return false or throw an exception.

(2)Check whether output is with the valid set of return values. In this case it’s not an issue because there are no “invalid” Boolean values.

2. Functional testing:

Ideally we should test all possible move ways in the board, but it’s too far big. So we can do a reasonable coverage of the ways. Assume there are 6 pieces in chess game, we need to write code like this:

1 for each piece a:

2 for each other type of piece b (6 types + empty space)

3 for each direction d

4 Create a board with piece a.

5 Place piece b in direction d.

6 Try to move – check return value.

11.4

To perform load testing, we must first identify the performance-critical scenarios and the metrics which we care about related to the performance. The typical criteria include respond time, throughput, resource utilization, maximum concurrence that the system can bear.

Then we design test cases to simulate the load, focusing on the measurement of these criteria.

In the absence of formal testing tools, we can basically create our own. For example, we can simulate concurrent users by creating thousands of virtual users. We can write a multi-thread program with thousands of threads, where each thread acts as a real-world user loading page. For each user, we could programmatically measure respond time, network throughput, etc.

We would then analyze the results based on the data gathered during the tests and compare them to the accepted values.

11.5

First we should make it clear that who will use this pen, what will they be doing with the pen, which type of the pen is, whether there are other requirements and so on. Then we can design proper test cases based on the information we just collected to test each point we care about.

11.6

The first thing we must do is to clarify assumptions about the ATM machine. We sould make clear following questions:

Who is going to use the machine? Anyone or blind people or other answers?

What service does the machine provide? “Withdrawing money”, “check balance”, “transferring money” or other answers.

What tools do we have to test? Do we have access to the code or just the machine?

Then we can design test cases for each service the machine provide. For example, to test the “withdrawing” function, we design test cases like this:

Withdrawing money less than the account balance

Withdrawing money greater than the account balance

Withdrawing money equal to the account balance

Withdrawing money from an ATM and from the internet at the same time

Withdrawing money when the connection to the bank’s network is lost

Withdrawing money from multiple ATMs simultaneously