1.What was the most useful technique you used to find the bugs? Why was it more useful thanother techniques you tried?

Utilizing print statements with printf seems to have been a helpful technique here. By printing out the values of variables at different points in my code, I can get a better understanding of what's happening.

2.Look up answers to the following questions and report your findings:

* 1. a.What are the largest integer and double values you can store?

The largest integer value I can store depends on the data type. For a signed 32-bit integer (int), the maximum value is 2,147,483,647. For an unsigned 32-bit integer, it's 4,294,967,295. As for double values, the maximum finite representable value is approximately 1.8 × 10^308.

* 1. b.Why is there a limit on the maximum value you can store in a variable?

There's a limit on the maximum value due to the finite number of bits used to represent the data type. For example, a 32-bit integer can only represent 2^32 different values.

* 1. c. If you exceed the maximum value an integer can hold, what happens? Explain why the format causes this to happen.
  2. I might experience overflow, which wraps the value around. For example, if I add 1 to the maximum signed 32-bit integer, it will wrap around to the minimum value.
  3. d. What is the format for the storage of a floating point variable? How does this differ from the way an integer is stored?
  4. The format for the storage of a floating-point variable typically follows the IEEE 754 standard. It consists of a sign bit, an exponent, and a mantissa. Unlike integers, floating-point numbers can represent a wide range of values, including fractions and very large or very small numbers.

3. What is the default amount of stack memory that is given to a program when Visual Studio starts a C or C++ program? What is the default heap size? Did you hit any of the limits? If so, which one(s)? If you hit a limit, would increasing the amount of memory allocated to the program fix the problem? Justify your answer. Why do they limit the stack and heap size for a program?

Visual Studio typically provides a default stack size of 1 MB for a C or C++ program. The default heap size is determined by the operating system. If your program is hitting memory limits, increasing the amount of allocated memory might help, but it's also crucial to manage memory efficiently to avoid potential issues like memory leaks.

The limits on stack and heap size exist for various reasons. Stack memory is limited because it's used for function calls, local variables, and other short-lived data. If it were too large, it could lead to stack overflow. Heap size limitations are often imposed by the operating system to prevent a single process from consuming excessive system resources. Allocating too much memory can lead to inefficient use of system resources and potential crashes.