Before we talk about programming, let’s consider some ideas that are worth thinking.

When and why were computers invented and developed?

You may have seen this photograph showing an early electronic computer.

Computers are not very smart. In fact, it was intentional making computers not smart, at least in the sense of our everyday life.

Computers were invented and developed during the Second World War and the cold war for military applications. In other words, computers were used to control weapons. Scientists and engineers did not want computers to think. Instead, they wanted computers to do exactly what computer programs instruct to do, nothing more and nothing less. Computers cannot tolerate some seemingly small or trivial mistakes because tolerating these mistakes means computers would be able to guess the mind of programmers. When computers were used to control weapons, it is dangerous if computers could guess the mind of programmers. This design philosophy has profound impact on how to write computer programs.

Today, computers are far smarter now and computers are used in many applications that do not involve human lives. However, you need to understand the history of computers so that you understand many challenges in developing computer programs.

Computer programs cannot tolerate any small or trivial mistakes because you do not tolerate small mistakes.

Let me give you a few examples. General Motor had a problem in ignition keys. About one hundred and fifty people were killed due to this problem. As a result, General Motor recalled about thirty million vehicles that potentially had this problem. If you calculate, one hundred and fifty people is a tiny fraction of thirty million. To put this in a different way, among the thirty million vehicles, most of them had not had this problem. However, that tiny fraction of failures is not acceptable.

Another example is the control system of Boeing’s 7 3 7 max airplanes. The airplanes have been grounded after two crashes, even though many thousands of flights were safely completed. Do you think the airplanes should be grounded after these two crashes, or the airplanes should be allowed to continue flying because many thousand flights were completed?

Do you think it is acceptable to say, “99.9% flights were safe and thus the crashes were acceptable”?

You may think this is an exceptional case. You may think the tiny fraction is serious because some people were killed. Let me ask you the following question. Imagine that you take a cross-continent flight. After the plane has landed, the pilot says, “Ladies and gentlemen, this is your captain speaking. We are delighted to inform you that we have flown more than ten thousand kilometers and landed safely. Our plane is now parked only ten meters away from the jet bridge. We have accomplished ninety nine point nine nine nine nine percent of the distance. For the remaining zero point zero zero zero zero one percent, please jump over the ten meters to the gate.

Thank you for flying with us. Please come again.”

What is your response? Would you accept that tiny fraction of zero point zero zero zero zero one percent of the distance? The plane has flown ten thousand kilometers.

You would not accept that, would you? In fact, you would not accept that if the plane is one meter away from the jet bridge. You want perfection. You demand perfection. You deserve perfection.

Computers are the same. Your computer programs cannot have any tiny trivial mistake. Computers do not know what to do if your programs have any mistake.

I did not set these rules. You did. You demand perfection, so do computers.

Let’s consider another example, in nineteen ninety four, Intel’s Penium processors had some problems. In some cases, the processors produced wrong results. There were different estimates about the seriousness of the problem. If you had bought this processor, would you accept the tiny fraction of possibility of mistakes, or you would demand a replacement? I am sure you would demand a replacement. You demand perfection, didn’t you?

You must accept the rules you have set: ninety nine point nine nine nine nine percent of success is failure. That is the rule you set to the companies. That is the rule by computers. If you sell a product that fails zero point zero zero zero one percent of time, would you be able to convince your customers that the product is acceptable? I wish you good luck because you would need it.

Software mistakes, also called bugs, caused huge financial losses worldwide. Estimates range from tens of billions of dollars to hundreds of billions of dollars. Many reasons may cause software mistakes. One of the reasons is the attitude that small mistakes are acceptable.

Computers are not smart and computers demand perfection. Do not expect computers to accept a program that has a tiny fraction of mistakes. Computers will not because they are designed this way.

Please understand that the real world does not have the concept of partial credits.

This class will give partial credits in order to encourage learning. However, you have to understand that this is an education environment.

This is not real world.

Do not get confused.

Next, consider another situation. You live in Pasadena California and you want to go to Boston. What would you do? If you are not familiar with the geographical locations, this is a map of the Los Angeles area.

This is a map of USA. California is at the West Coast and Boston is at the East Coast.

You have many options. The first is to start driving east because Boston is east of Pasadena. The second option is to start walking east.

The third option is to drive west to the Los Angeles Airport and take a flight.

Which one would you choose? Why?

I would drive west to take the flight. How about you?

Sometimes, the shortest route needs a detour. If you choose the first two options, you will be closer to Boston after one or even two hours. If you choose the third, you will spend time going to the airport, wait to pass security checking, and wait for boarding. After one or even two hours, you are still at the airport.

However, if you choose to fly, you will be in Boston a few hours later. If you drive or walk east, you will be closer to Boston after one or two hours but you will get to Boston after several days by driving or many weeks by walking.

What is the point?

For long distance traveling, flying is a much better choice. An airport is a much better tool than a car or walking. To accomplish your goal, you need to choose the right tool. In this case, an airplane is the right tool.

Sometimes, the shortest route needs a detour. When you learn computer programming, you need to know many tools. Learning these tools takes time. However, after you learn these tools, you can save a lot of time later. I have worked with many students in computer programming. Some of them do not want to spend time learning good tools. They spend much more time later because they use wrong tools.

This is another important concept. You need to know the differences between solving a problem and fixing a solution. Imagine that your roof leaks and it is raining, you take a bucket to catch the water but the bucket has a hole.

What will you do when it is sunny?

You can fix the roof so that it does not leak.

Or, you can buy a new bucket so that it can catch water on the next rainy day.

Which would you do?

You may think the answer is obvious. It is not. The first is to solve the problem. If you fix the roof and it does not leak, you do not need the bucket. The second is to fix a solution. The roof still leaks but you catch water using a good bucket.

Why is it important to distinguish these two? I have observed many students trying to fix their programs. This is also called debugging. Sometimes, however, the solutions are wrong and these students spend a lot of time trying to fix the solutions.

You need to know whether you are solving a problem or fixing a solution. They are different.

These are some ideas worth thinking before you write another line of computer programs.

Now, welcome to the world of computer programming.