Introduction to Problem Solving and Programming

Prof. Deepak Gupta

Department of Computer Science Engineering

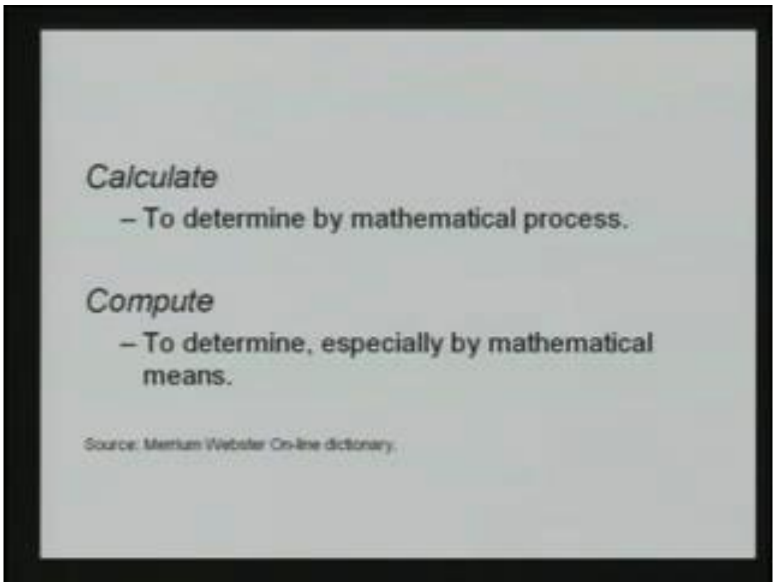
Indian Institute of Technology, Kanpur

Lecture – 1

The title of the course is Introduction to Computing. In this course, we are going to study how to use computers effectively for solving real life problems that we will come across in your professional career. As you know in today's world, the computer is an invaluable tool to every engineers and scientist for solving problem related to his or her discipline. For example, if you are a civil engineer, you need a computer to compute the statics on the various structures of the building or if you want to be an aerospace engineer, and you might need use the computer to compute the airflow around an aircraft. So, it is important for all of us to understand how to use computers effectively to solve our problems and not as the users of the computers, we will be able to tell the computer that how to solve our problem.

So, in this course, we are going to basically see how to use computers to effectively solving engineering problems; and as the first step, we will try and understand what a computer is and at very high level of what it tell. So, what is a computer and as the word suggest it something that compute. Well index is how it is different from a simpler device that you are all familiar namely with the calculator, a calculator calculate. So, the words calculate and compute are different.

(Refer Slide Time: 01:52)



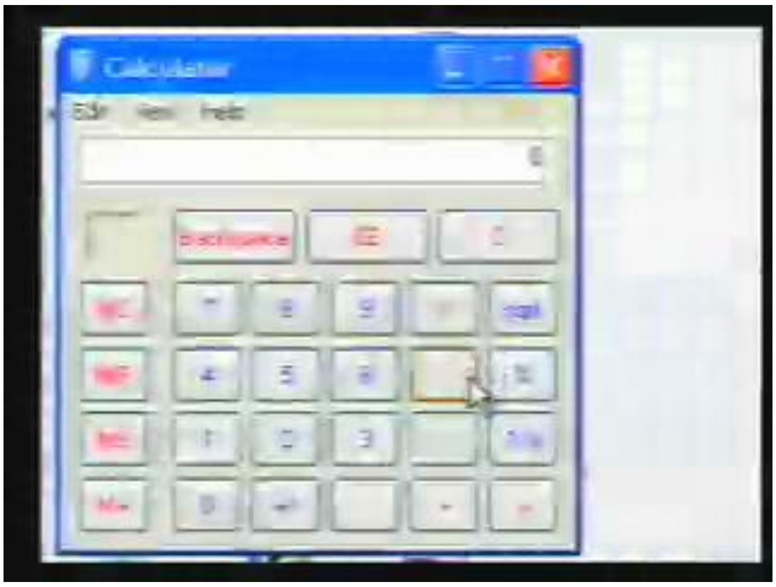
Calculate
- To determine by mathematical process.

Compute
- To determine, especially by mathematical means.

Source: Merriam Webster On-line dictionary.

Let us try in consult dictionary to find out here is what calculate means to determine something by mathematical process, and here is what compute means to determine especially by mathematical means. So, we can see the meanings of two words are quiet similar. And therefore, one would expect a calculator to be similar to a computer, but exactly not the case a computer has a rounded very soon, it is much more powerful and we can use it in much more flexible and interesting ways than we can use it calculator.

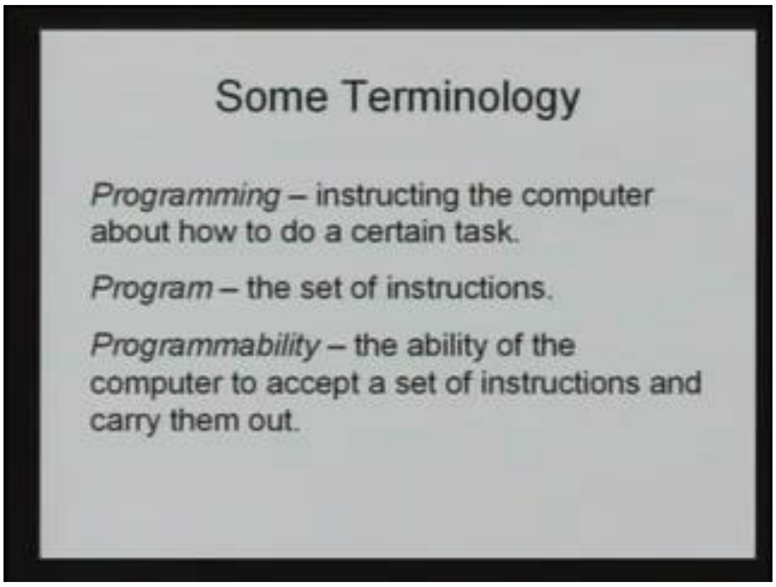
(Refer Slide Time: 02: 37)



Now we are going to see what you can do using a calculator something that is familiar to all of you. Let me show you a calculator. So, here is what is standard calculator looks like. As you can see, it has button for various primitives arithmetic operations, and you can use these operations to do some calculations with number. For example, 56 into 23 is 1288; otherwise, what we cannot use this calculator to do is to tell it how to combined these operations in a certain fashion, so that a certain computation can be achieved. For example, how do I compute let us say a factorial of a number using this calculator. You might say that is very easy. If I have to calculate the factorial of five then what I will do five times four times three times two that one twenty is the factorial of five and that correct, but you cannot take this calculator how to compute the factorial of any given number. For any specific given number you can use the calculator to find the calculator by combining these operations in a certain way that you know, but it cannot hit the calculator, how to the combine these operations in some way, so that it can compute the factorial of any number automatically.

We might say if I want to do that I think using scientific calculator which has the primitive operations for factorial, but even the scientific calculators the set of operations that you have is fixed and you still cannot tell it how to combine these operations automatically in certain sequence, so as to achieve a certain computation. For example, your scientific calculator might have button for factorial, but for example, let us say the area of the intersection of the two given circle will look at and that will be that important difference between the calculator and computer, where the computer can give actually select how to combine the various primitive operations that are available in a certain fashions, so as to achieve the desired computational to solve the desired problem.

(Refer Slide Time: 05:13)



Some Terminology

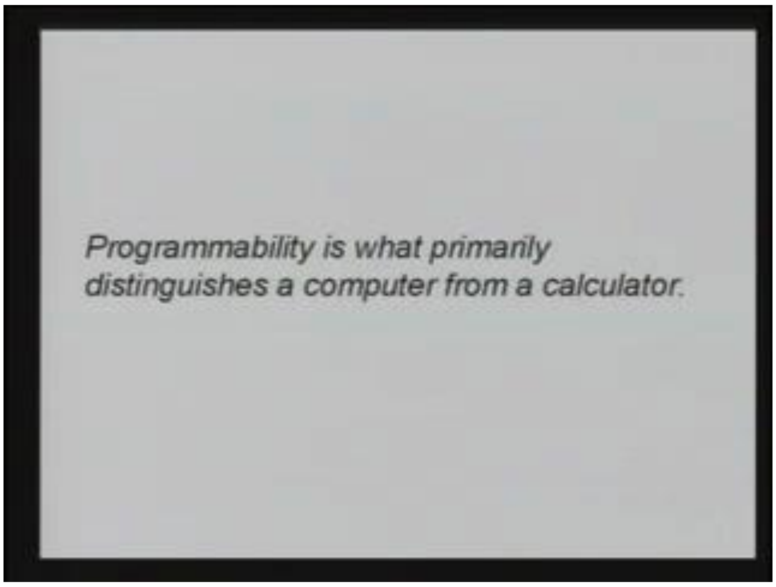
Programming – instructing the computer about how to do a certain task.

Program – the set of instructions.

Programmability – the ability of the computer to accept a set of instructions and carry them out.

Let us look at some terminology now, which relate to the computer that are talking about. It is instructing the computer about how to do a certain task is what is called the programming tutorial come cross this term before. A program is nothing but the set of instructions that you will give it to the computer to quite carry out certain task; and the third term is programmability, which is the ability of the computer to accept set of instructions from you and to face fully carry them out.

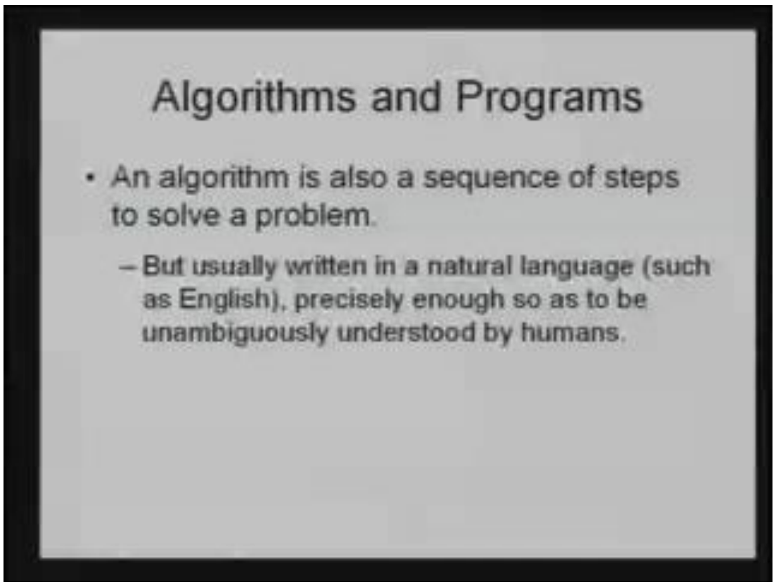
(Refer Slide Time: 05:36)



Programmability is what primarily distinguishes a computer from a calculator.

And this programmability is what primarily distinguishes a computer from a calculator because a computer is programmable that is we can give it to set of instructions to solve the problem of archive file where as we cannot program a calculator or we cannot instructed how to carry out the certain task. Of course, there are other important differences between computers and calculators, which are also significant. For example, the calculator can only deal with numbers where as the computer can deal with data in variety of forms as some of you might be aware. For example, it can it can handle text, it can handle images, it can handle sound, and it can take input from the variety of interrupting input devices and so on, but the fundamental difference again is that a computer is programmable while a calculator is not. I already said the program is the sequence of instruction set to computer in order to solve our problem at end there is a related term called an algorithm, which is similar to the program, but different in certain details.

(Refer Slide Time: 06:45)



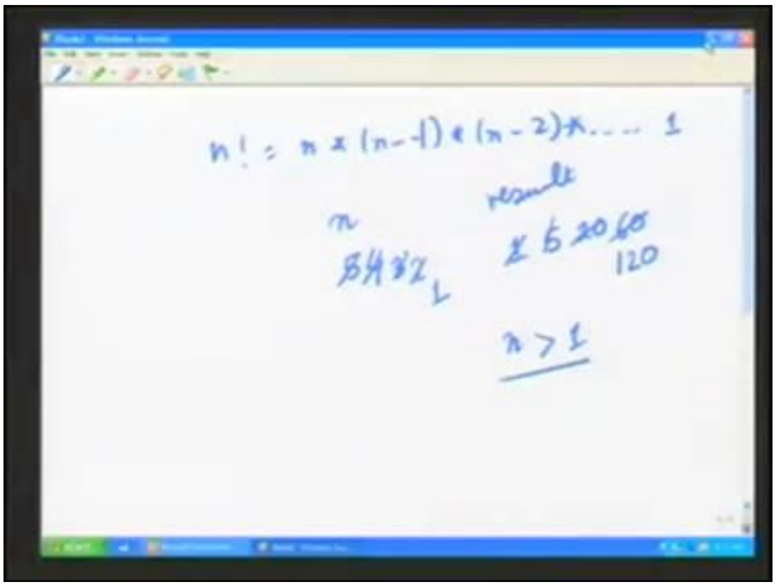
Algorithms and Programs

- An algorithm is also a sequence of steps to solve a problem.
  - But usually written in a natural language (such as English), precisely enough so as to be unambiguously understood by humans.

So, an algorithm is also a sequence of steps to solve a problem, but it usually written in a natural language such as English and written precisely enough so as to be unambiguously understood by humans. A program on the other hand is an essential algorithm translated to what is known as the programming language, that computer can understand that attitude that is the fundamental difference between algorithms and programs. You might think of an algorithm as a solution to the problem, whereas the program is the translation of the solution from English to a programming language that a computer can understand.

In this course, we are going to talk both about how to develop algorithm for solving some simple problem, which we will find useful and building blocks for solving more complex problem; and at the same time, we also going to see how to express the algorithm in a programming language, and actually execute the program on a computer. So, let us now take an example let us say we take up old example of a computing factorial and you want to write an algorithm to compute the factorial of the given number n. Note that the n could be any number and we do not know a prime what is the number n is. So, at sequence of step namely our algorithm to find the factorial of n should be such like to handle any given n and not just a specific n. So, how do we go about finding such an algorithm. We all know how to compute factorial.

(Refer Slide Time: 08:40)

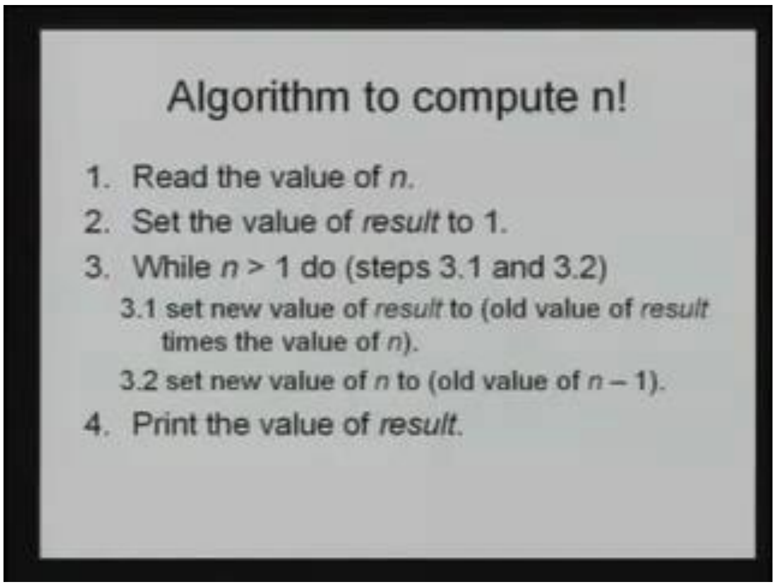


$n! = n \times (n-1) \times (n-2) \times \ldots 1$
result
n
$\cancel{5}\cancel{4}\cancel{3}\cancel{2}1$
$\cancel{1}\,\cancel{5}\,\cancel{20}\,\cancel{60}\,120$
$n > 1$

So, we know the factorial of n is nothing but n times n minus 1 times n minus 2 times till 1, but how do I express this as an algorithm moved that multiplication is fundamentally a binary operation and we can really only multiply two numbers at a time right. So, how do we go about converting this knowledge of all about the definition of factorial into an algorithm for the factorial well let us think a little. As you can imagine what you can do if you want to multiply a numbers two at a time, we can start with one let us call it for time being as the result and is whatever it is and in each step first let us say multiply result by the value of n that is the first step. I multiply result by the value of n that we give us n.

Select an example, let n can be five, so start with the result as one; in the first step, we multiply the value of the result by the value of n which results five and that new value of the result. And the next step, you have to multiply the result by four not five therefore, we decrease the value of n by one and it is four. And in the next step, we do the same thing, multiply the result by the new value of n and that we get twenty decrease the value of n by three, and we keep doing this. So, in the next step, twenty become sixty and n becomes two, and next step sixty becomes one twenty and n becomes one and how long we keep doing this as long as n is greater than one. So, once you have done this, we have essentially arrived at an algorithm to compute the factorial. All we have to do is now express our thoughts more precisely.

(Refer Slide Time: 10:41)



Algorithm to compute n!

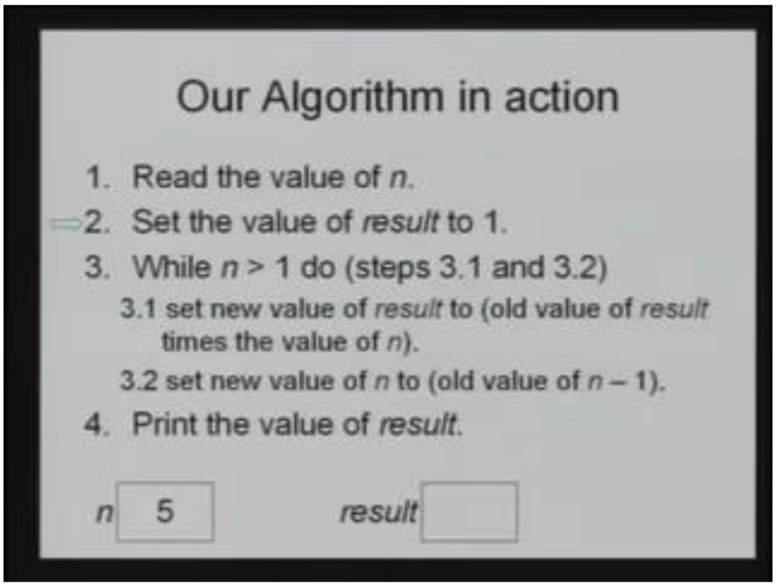
```
1. Read the value of n.
2. Set the value of result to 1.
3. While n > 1 do (steps 3.1 and 3.2)
    3.1 set new value of result to (old value of result
        times the value of n).
    3.2 set new value of n to (old value of n – 1).
4. Print the value of result.
```

So, here is the expression of our thoughts in the form of an algorithm. So, there is the algorithm to compute the factorial of n. The first step is read the value of n; and in the second step, as I said earlier reversibly, the value of result initially to one and as long as n is greater than one keep doing these two steps. And these two steps, as we seen are multiply the value of the result by the value of n and decreasing the value of n minus one. So, set the new value of the result to the old value of the result times the value of n, and the second step is new value of n to old value of n minus one, and keep doing this as you can see here since as along as n is greater than one. And once it is done, the value of the result continuously answer to our problem namely the value of n factorial, and we can output the value of the result in this algorithm right now we are not really bothered too much about z y read this value of n form or zy print. The value of the result how do I print the value of the result that is not something bothering it is the step two, three - 3.1, and 3.2 which are really important at this stage.

And let us note some more things about this algorithm, note that our instruction for computing the factorial of n is just not is checked in the sequence of instruction to be executed by the computer one by one, but there are things which we have repeatedly done. The step three point one and three point two have to be repeatedly perform as long as the certain condition hold, this is called a loop in technical jargons. The second thing that you should note is that you have to remember the result of intermediate computation, and the intermediate computation that you can seen this particular simple example at two namely the value of the results, and the value of n.

Since these values you have to remember is called as variable. So in this examples, there are two variables in this algorithm, result and n. I am sure the notion of variables is familiar to you from your high school algebra where you have equations like x plus y is equal to x square plus y square something like this, but that mathematical notion of variable is very different from the programming notion of variable. In mathematics, variables such as x and y denote unknown, but fixed values; whereas in an algorithm variable denote the values that we change over time. As you can see the values of the result and n change over the time as we proceed with the algorithm and execute it step by step and that is what the programming notion of the variable is.

(Refer Slide Time: 14:00)



Our Algorithm in action

```
1. Read the value of n.
⇨2. Set the value of result to 1.
3. While n > 1 do (steps 3.1 and 3.2)
   3.1 set new value of result to (old value of result
       times the value of n).
   3.2 set new value of n to (old value of n – 1).
4. Print the value of result.
```

n 5 result

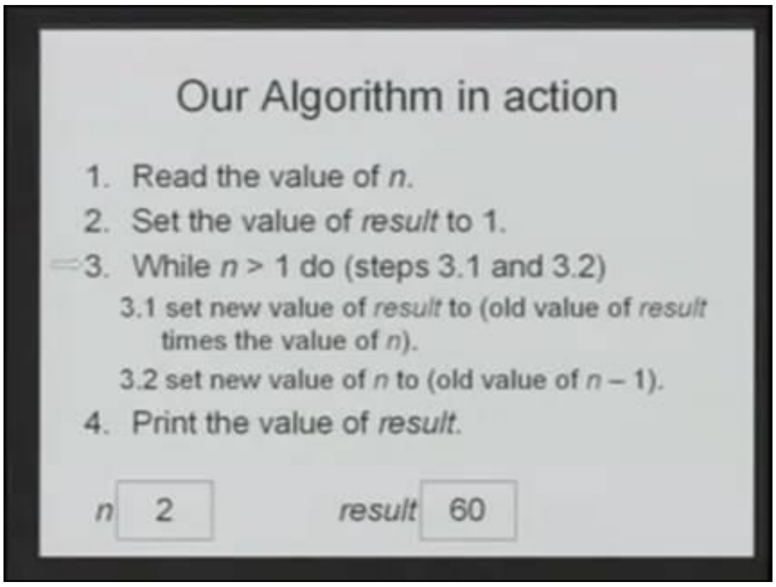
So, let us now see this algorithm in action. So, here is the same algorithm and let us see step by step how it is going to execute and compute the desired results. Here the first step read the value of n. Let take an example, the same example I taken it before, and let us assume the value of n that is given to us is five. This blue arrow indicates where is the execution we are currently right. So, up to having executed line number one we now have to execute line number two and the result of executing the line number one was the value of n has become five.

So, the second step is set the value result to one and what you think this will do as you imagine this will do nothing but set the value of the result to one here it is. Now, third step something this is the step which is to be repeated as long as the condition hold while n is greater than one do the sequence of steps. So, this particular line in the algorithm in the sense just going to check whether or not and it is greater than one. The current value of n is five exactly is greater than one and therefore, the answer to this condition are the result of the condition is true. And therefore, we have to go for the step three point one and three point two and before the condition evaluated to true we have to come to the step three point one which says set the new value of the result to the old value of the result times the value of n. So, the new value of the result will be result times n which is the five times one which is the five in that course it is line in the algorithm does.

The result becomes five note that in this step n does not change. In the next step, you have to set the value new value of n to old value of n minus one, this is five minus one namely four. So, we do that and note that once we have finished executing the three point one and three point two. We must come back to line number three and check this condition again, why because remember that the step three point one and three point two has to be repeated as long as this condition hold. And this condition will not as you can imagine cant to be hold for ever because in every step you are decreasing the value of n by one and therefore after sometime certainly after finite number of steps, the value of n will become less than equal to one at which time we will stop repeating the step 31. and 3.2. But at this point of time, the value of n is still greater than one.

So, we repeat the same step, you come here the value result now become twenty five times four, and now the value of n becomes three we go back and still more than one, and therefore, we repeat the step again. So, the result now become sixty and n now becomes two, which is still greater than one. So, we do it once more result now becomes one twenty the value of n becomes one. Now when we check whether n is greater than one or not we conclude that n is not greater than one because n is equal to one.

(Refer Slide Time: 17:02)



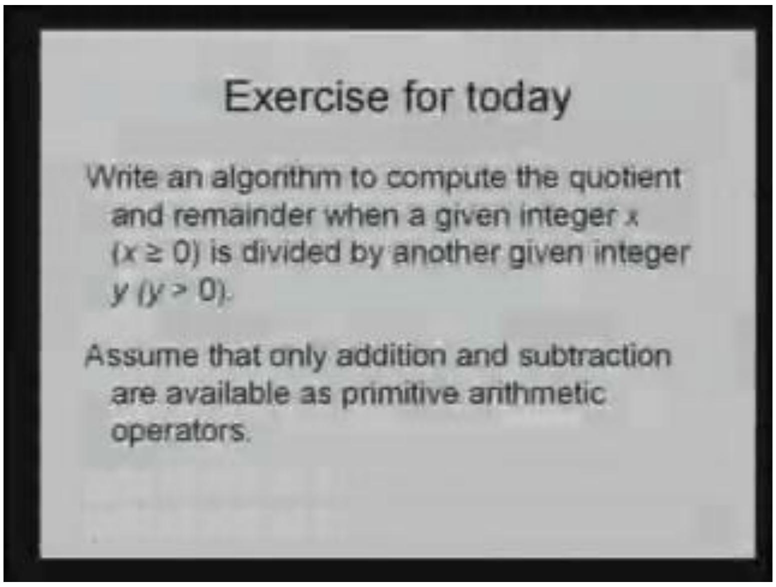
Our Algorithm in action

```
1. Read the value of n.
2. Set the value of result to 1.
3. While n > 1 do (steps 3.1 and 3.2)
    3.1 set new value of result to (old value of result
        times the value of n).
    3.2 set new value of n to (old value of n – 1).
4. Print the value of result.
```

n 2 result 60

And therefore, this condition is no longer satisfied now; and therefore, the step three point and three point two will not be executed now, we straight away come to step four which is to print the value of the result. And once you do that the output here the value of the result which of course is one twenty and which of course, we will know it is the factorial of n. And we could have solve the same problem in other ways as well, for example, instead of multiplying one by n and then by n minus one and then by n minus two, we could have started this process from the reverse direction that is multiply one by two and then by three and then by four and so on. You could express that algorithm in this fashion in similar to what we have done here, and I leave that as an exercise for you. And at the end of the lecture carries one more exercise for you to try yourself.

(Refer Slide Time: 18:23)



Exercise for today

Write an algorithm to compute the quotient and remainder when a given integer x ($x \geq 0$) is divided by another given integer y ($y > 0$).

Assume that only addition and subtraction are available as primitive arithmetic operators.

I want you to write algorithm to compute the quotient and the remainder when a given integer x which is more than or equal to zero is divided by another given integer by y which is known to be greater than zero. So, for example, if let us say fifteen is divided by seven then the quotient should be two and the remainder should be one, but assume that only addition and subtraction are the arithmetic operations available to you. So please write an algorithm for doing that just to reinforce the concept you learned today. Before we end this lecture today, let me summarize what we have seen so far. We have seen the notion for the programming the program and programmability and this notion of programming and programmability is essentially as we have seen what is the computer for because we can write programs for our problems, and tell the computer to say to find the result that we need.

And before we actually writing the programs and programming languages are computer can understand, we have seen what is the notion of an algorithm which is independent of any programming language. And we saw a simple algorithm to solve simple problem namely to find the factorial of the given number n and we saw how the computer would go about executing this algorithm.