

# AP Computer Science Principles Ultimate Guide

## Chapter 1: Creative Development

### Collaboration

- **Programming** is a collaborative and creative process that brings ideas to life through the development of software.
- **Software development processes** used in the industry often require students to work together in teams.
- Collaboration can take place at various points and in various ways.
- An important concept that is discussed throughout the entire course is the idea of computing innovations.
- A **computing innovation** uses a computer program to take in data, transform data and output data.
- **Collaboration** can occur in the planning, designing, or testing (debugging) part of the development process.
  - Collaboration tools allow students and teachers to exchange resources in several different ways, depending on what suits a particular task.
- **Collaborative learning** can occur peer-to-peer or in larger groups.
- Peer instruction involves students working in pairs or small groups to discuss concepts to find solutions to problems.

### Identifying and Correcting Errors

#### There are 4 main types of errors in programming:

- **Syntax Error:** A mistake in which the rules of the programming language are not followed.
- For Example:

```
a ← expression
```

```
DISPLAY (A)
```

A syntax error in this example occurs because the second statement attempts to display the variable A, which is not the defined variable. Variable names are case sensitive. Therefore, while the variable with a lowercase a in lowercase is defined by the first statement, the variable with a capital A in the second not defined. Therefore, the rules of the programming language were violated

- **Runtime Error:** A mistake that occurs during the execution of a program that ceases the execution.

- For example:

DISPLAY (5/0)

In this example, there is no syntax error, because the language of the code is used correctly. However, this causes a runtime error because you cannot divide by zero. The execution of the program will halt at this line.

- **Logic Error:** A mistake in the algorithm or program that causes it to behave incorrectly or unexpectedly.

- For example:

a 95

IF (a > 90)

DISPLAY("You got an A.")

IF (a > 80)

DISPLAY("You got a B.")

IF (a > 70)

DISPLAY("You got a C.")

- The code is intended to correctly print out what grade the student got. Since this particular student's score was a 95, which is greater than 90, the program should display You got an A. However, the program actually prints out the following:

You got an A.

You got a B.

You got a C.

This logic error occurs because the student's score is also greater than 80 and 70 with no restriction that prevents multiple grades from being printed.

- **Overflow Error:** A mistake that occurs when a computer attempts to handle a number that is outside of the defined range of values.

- For example:

x = 2000\*365

DISPLAY (x)

The result of the multiplication is a large number. In many languages, this product would be large enough to be outside the range of certain data types. Therefore, if x is defined as a variable of one of those data types, this multiplication will cause an overflow error.

- **Debugging:** is the process of finding and fixing errors.
  - You should use test cases, extra output statements, examination for syntax errors and other debugging tools to find and fix any errors.

## Chapter 2: Data

### Binary Numbers

- Any digital data has a numerical representation using binary numbers.
- A **bit** is the smallest unit of information stored or manipulated on a computer; it consists of either zero or one.

## Base Conversion

### Binary to Decimal Conversion

- Of course, binary numbers are rarely used in real life.
- Therefore, programmers must be able to go back and forth between the binary numbers we use in computing and the decimal numbers that we use in everyday life.
- The key is to remember that the different binary digits represent different powers of 2.
- For example, let's use the binary number 1101.

Use this table:

$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$	
128	64	32	16	8	4	2	1	
0	0	0	0	1	1	0	1	
0	0	0	0	$8 \times 1$	$4 \times 1$	$2 \times 0$	$1 \times 1$	
0	0	0	0	8	+ 4	+ 0	+ 1	= 13

Therefore,  $(1101)_2 = (13)_{10}$ .

### Decimal to Binary Conversion

- We need to find the powers of 2 that add up to the given decimal number. Start by finding the largest power of 2 that is less than the number.
- Subtract that number from the original, and repeat until you're down to 0.
- Try the example of the decimal number 200.

Use this table:

$2^8$	$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$
256	128	64	32	16	8	4	2	1
0	1	1	0	0	1	0	0	0

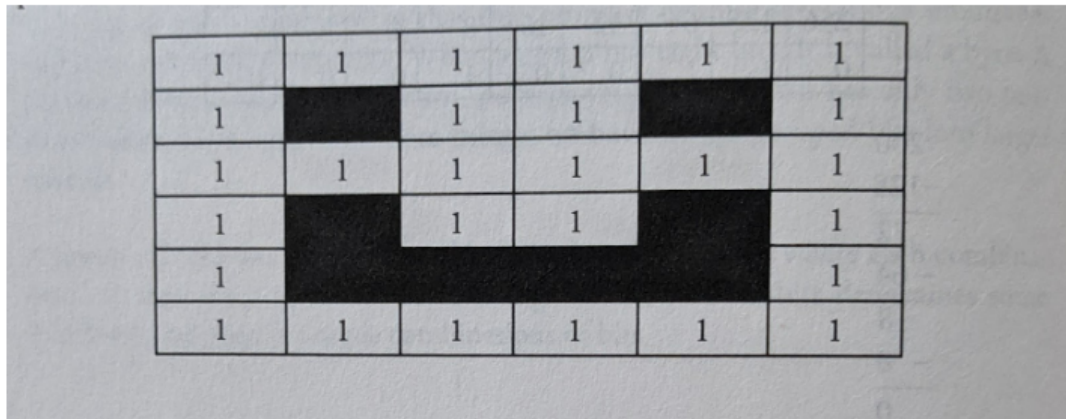
  

200
-128
<hr/>
72
-64
<hr/>
8
-8
<hr/>
0

Therefore, the decimal number 200 is equivalent to the binary number 1100 1000.

## Digital Images as Bits

- Images displayed on the screen are converted into binary formats and then processed by a computer displayed on our screen.
- **Digital images:** are a collection of pixels. where each pixel consists of binary numbers.
- If we say that one is black (or on) and 0 is white (or off), then a simple black and white picture can be created using binary Draw a grid and color the squares (1-black and 0-white) to create the picture
- However, before creating the grid, the size of the grid needs to be known.
  - This data is called **metadata**, and computers need metadata to know the size of an image.
  - The metadata for the image to be created is 10 x 10; this means the picture will be 10 pixels across and 10 pixels down.



0	0	0	0	0	0
0	1	1	1	0	0
0	1	1	1	0	0
0	1	1	1	0	0
0	0	0	0	1	1
0	0	0	0	1	1
0	0	0	0	0	0

## Binary and Color Representation

- **Images:** are not often just black and white.
- To represent colors computers also use binary numbers.
- **Color:** is based on light.
- Any color can be created using red, green and blue light.
- The maximum value for any color in decimal 255, which is represented by 11111111 in binary.
- The minimum number is 0.

Here are some examples of color representation showing both the decimal values and the binary values:

White = [ 255, 255, 255 ]	[11111111,11111111,11111111]
Black = [ 0, 0, 0 ]	[0,0,0]
Blue = [0,0,255]	[0,0,11111111]
Red = [255, 0, 0]	[11111111,0,0]

## Music as Bits

- An **analog signal** exists throughout a continuous interval of time and takes on a continuous range of values.
- A **digital signal** is a sequence of discrete symbols.
  - If these symbols are zeros and ones, we call them bits.

- As such, a digital signal is neither continuous in time nor continuous in its range of values.
- **Sampling:** is recording an analog signal at regular discrete moments and converting them to a digital signal.
- Digital signals are resilient against noise.

## Data Compression

- **Data compression:** is used everywhere.
- Mp3, mp4, rar, zip, jpg, and png files (along with many others) all use compressed data.
- **Compression:** is also an important consideration when it comes to backing up and archiving your important files, particularly for uploading over the Internet.
  - Compression is a two- way process: a compression algorithm can be used to make a data package smaller, but it can also run the other way, to decompress the package into its original form.
- Data compression: is useful in computing to save disk space, or to reduce the bandwidth used when sending data (eg, over the Internet).
- Data compression deals with taking a string of bytes and compressing it down to a smaller set of bytes, whereby it takes either less bandwidth to transmit the string or to store it to disk.
- **Lossless algorithms:** are those that can reconstruct the original message exactly from the compressed message, and lossy algorithms can only reconstruct an approximation of the original message.
  - Lossless algorithms are typically used for text, and lossy algorithms for images and sound where a little bit of loss in resolution is often undetectable, or at least acceptable.
- **Lossless compression:** packs data in such a way that the compressed package can be decompressed, and the data can be pulled out exactly the same as it went in.
- **Text compression:** is another important area for lossless compression.
  - It is very important that the reconstruction is identical to the original text, as very small differences can result in statements with very different meanings.
- **Lossy compression** is a technique that does not decompress digital data back to 100% of the original.
  - Lossy methods can provide high degrees of compression and result in smaller compressed files, but some number of the original pixels, sound waves, or video frames are removed forever.
  - Lossy is used in an abstract sense, however, and does not mean random lost pixels, but instead means loss of a quantity such as a frequency component, or perhaps loss of noise.
- **Images:** high image compression loss can be observed in photos when enlarged
- **Music:** there is a difference between an MP3 music and a high-resolution audio file



- **Video:** moving frames of video can handle a greater loss of pixels compared to an image

## Using Programs with Data

- The increase in digitization of information, mixed with multiple transactions, has resulted in a flood of data.
- The advancement in technology has promoted the rapid growth of data volume in recent years.
- By analyzing large data sets of data, it is possible to categorize connections from unconnected data sources and find specific patterns.
- **Data extraction:** is the process of obtaining data from a database or software such as a social media website so that it can transport it to another software (such as spreadsheets) designed to support online analytical processing.
- Data extraction is the first step.
  - The next step is to transform (either through filters or programs).
  - The final step is to analyze using graphs and other data visualization tools.

## Below are the steps to extract data and analyze them:

- Analyze the data sources.
  - Data sources are found in different forms like web pages, emails, and chat video files, audio files, text documents, customer messages.
- Know what will be done with the results of the analysis.
  - It is vital to understand what sort of outcome is required.
  - Is it a trend, effect, cause, quantity, or something else that is needed?
- Decide the tools needed to read the data, and the repositories such as databases needed to store the data.
  - Clean the data of whitespace, symbols, duplicates, etc.
  - Understand the data patterns and text flow. This should be done using visualization tools.

## How to read and analyze graphs

- A **graph** is a pictorial representation, a diagram used to represent data.
  - It usually is used to depict a relationship.
  - **Graphs and charts:** represent data in points, lines, bars, pie charts, and scatter plots.
  - Different types of graphs and charts display data in different ways.
  - Some are better suited than others for different uses.
- **Picture graphs:** use pictures to represent values.
- **Bar graphs:** use either vertical or horizontal bars to represent the values.
- **Line graphs:** use lines to represent the values.

- **Scatter plots:** represent the data with points, and then a best-fit line is drawn through some of the points.

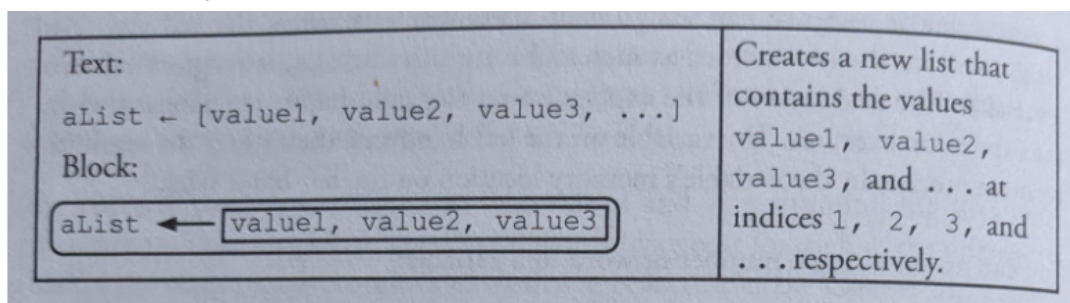
## Chapter 3: Algorithms and Programming

### Algorithms

- **Algorithm:** is a set of steps to do a task.
- Recipes are algorithms.
  - They are a set of steps to prepare food.
- Instructions for taking medicine make up an algorithm.
- Directions to get from one location to another are an algorithm.
- **Computer science algorithms:** are the set of steps to solve a problem or complete a task.
- **Algorithms:** are implemented with software.
- Examples are programs to:
  - Calculate grade averages
  - Run the air conditioner when the room temperature reaches 78°F
  - Calculate the shortest route from home to school on your GPS
- Algorithms have the potential to solve many problems once the programs are written for them to run.
- **Section of code:** may work independently or can be used with other programming modules.
  - These program modules read in values, make computations as needed, and produce output to automate processes.

### Data Abstraction

- A popular way of defining abstraction is information hiding.
- Just as related program statements are bundled together, related program variables can be bundled together.
- Such abstractions allow us to think of the data within a program hierarchically.
- A list is an example of data abstraction.



- A list is a data type that holds a collection of values.
- aList[3, 7, 11]  
aList is described as a "list of integers."

Within a list, when accessing its parts using an integer index, `aList[1]` gives us the value 3, `aList[2] = 7`, and so on. Lists allow for data abstraction in that we can give a name to a set of memory cells. For instance, in a `colorList`, a list that holds three colors `['red', 'blue', 'green']` instead of using three separate variables, `color1`, `color2`, etc., one variable `colorList` holds all the three variables. Each of the contents of the list is accessed by changing the index value.

## Variables

- **Variables:** are placeholders for values a program needs to use.
- A program may need variables for numbers, both integers and real numbers, as well as text fields and Boolean values.
- Variables can be used in expressions, both mathematical and textual.
- An important aspect of variables is naming them well.
- Variables are data abstractions because we do not need to know the details of how and where the values are stored in memory.

## Strings

- **Strings:** are text fields that are just a series of characters and are denoted with quotation marks around the string field.
- Any character, number, or symbol can be part of a string.
- When numbers are part of a string, then they will be considered to be text, like in a street address.
  - In this case, you cannot use them in calculations because they are text fields, not numbers.

## Assignment Statements

- To assign a value to a variable, the assignment operator is used.
- In many programming languages, a single equals sign, `=`, is used as the assignment operator.
- The variable name is always on the left of the `←` sign.
- The programming language will evaluate the right hand side of the assignment operator and then place the value into the variable on the left-hand side of it.

Text :

```
variable ← expression
```

Block:

```
variable ← expression
```

- The previous value stored in a variable is overwritten by a new assignment statement.
  - In this example, a variable named `score` is assigned the value 10.



- It is then assigned the value 11.
- The value 10 is gone and no longer available.
- **Expressions:** are calculations to be evaluated to an answer or single value.
- In computer science, the value is then assigned to a variable if the value will be needed later.
- It also could be displayed on a screen or printed.

<code>x ← 3.14 * 5</code>	Multiply 3.14 by 5 and store it in the variable x.
<code>PI ← 3.14</code>	Assign a variable named PI the value 3.14.
<code>area ← PI * radius * radius</code>	Assign the variable area the result of PI times radius <sup>2</sup> .

- You will get an error if you write:

`5/9 * (fahr - 32) ← celsius`

- The variable must be on the left and the assignment arrow points left.

<code>celsius ← 5/9 * (fahr - 32)</code>	This is the correct way to set up the calculation.
--	--

- The expression will be evaluated and loaded into the variable celsius.
- P Parentheses
- E Exponents
- M Multiplication
- D Division
- A Addition
- S Subtraction

## Boolean Values

- **Boolean values:** are one of the foundations of computer code.
- Understanding Boolean is important for writing correct, readable, and efficient code.
- Boolean values can only be true or false, needing only one bit to represent its value.
- Relational operators are used with Boolean values.
- These are just like in your math class:

**= equal**

**≠ not equal**

**> greater than**

**< less than**

**≥ greater than or equal to**

**≤ less than or equal to**

# Program Statements

- All programs can be written using a combination of only three types of statements.
- These statements are used over and over and can be combined for more complex needs, but there are only three.

## Sequential

- These are statements that are executed as written in order in the program.
- As we start writing our programs, a code statement has an action that will be executed when the program is run.
- Once a statement is complete, the next statement is run.
- Naturally, they have to be in the correct order to accurately complete the algorithm.

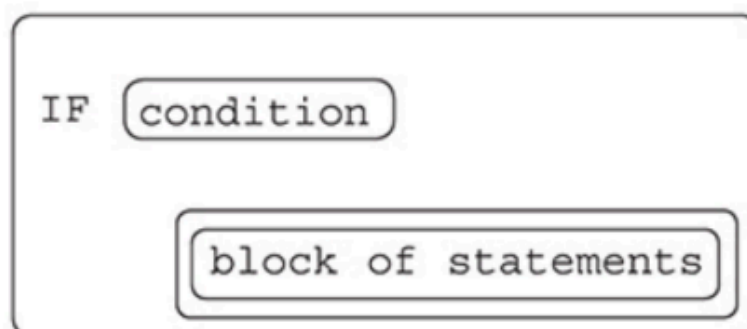
## Selection Statements

- These are a key component to many programs.
- They use the "if (condition)" structure, and the evaluation of the condition uses Boolean values.
- As we know, Booleans can only be either true or false, so the program statements associated with the condition only run when the condition at that moment evaluates to "true."
- This changes the flow of the program from every statement running sequentially to filtering out some of the code that will execute, based on the Boolean value.

Text :

```
IF(condition)
{
    <block of statements>
}
```

Block:



- Notice that the code to be executed when the condition is true is surrounded by the curly braces, { }, in the "Text" version.
- The code within the braces is indented, which helps with readability.
- It also makes it easier to see if you are missing a closing brace }!

## Iterative

- **Iterative statements:** are also referred to as repetitive statements or loops.
- This type of programming statement also changes the sequence of lines of code that are executed.
- Code that is associated with these statements repeats as many times as specified before continuing with the rest of the program.

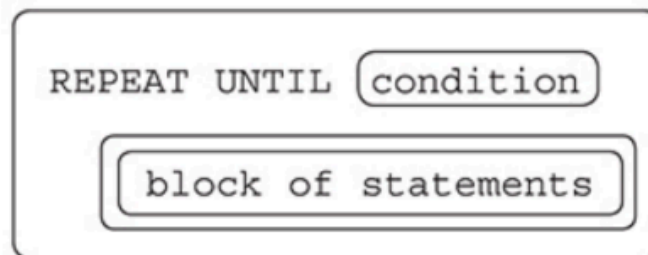
## Repeat Until (Condition) Loop

- The REPEAT UNTIL loop has a condition to evaluate at each iteration of the loop.
- The loop will continue to run while the condition evaluates to "false."
- This is similar to how an IF statement works except the condition for the IF statement must evaluate to true for it's code to execute.

Text:

```
REPEAT UNTIL (condition)
{
    <block of statements>
}
```

Block:



- **Combining Algorithms:** One of the key features of algorithms is that once they are created, you can use them over and over, combine them for more complex problem solving, or modify them for a new use.

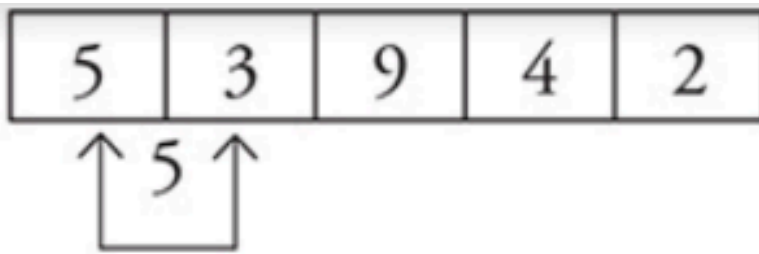
## Nested Conditionals

- In a program or code snippet where there is an IF statement within another set of IF statements, these are called nested IF statements.
- When the outer IF statement is executed, the inner IF statement may also get executed.
- This allows for the program solution to evaluate another expression after determining the results of a previous decision.

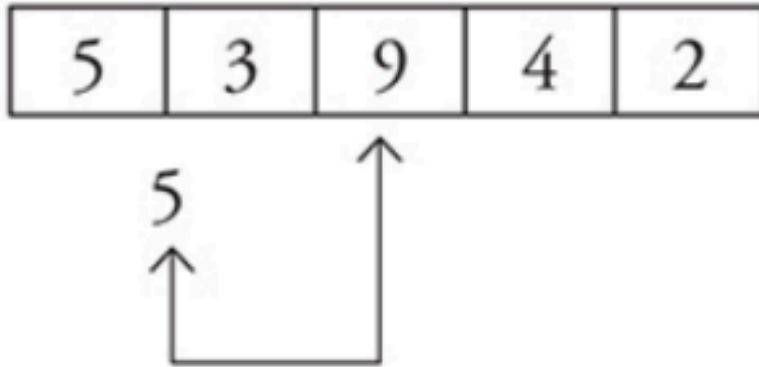
```
IF (number1 < number2)
{
    IF (number1 < number3)
    {
        smallerNumber ← number1
    }
    ELSE
    {
        smallerNumber ← number3
    }
}
ELSE
{
    IF (number1 < number3)
    {
        smallerNumber ← number2
    }
    ELSE
    {
        smallerNumber ← number3
    }
}
```

## Common Algorithms

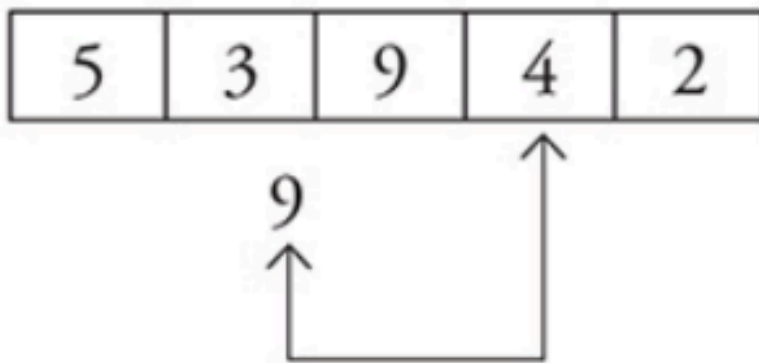
- Determining the maximum or minimum number from two or more numbers.
- The pseudocode for the maximum of two numbers is:
  - Compare the numbers.
  - Store the larger number as the maximum.
- For the minimum of two numbers:
  - Compare the numbers.
  - Store the smaller number as the minimum.
- If you have more than two numbers, a computer can still only compare two at a time.
- One number would be the current largest (or smallest), and the other would be the next number to compare it to.



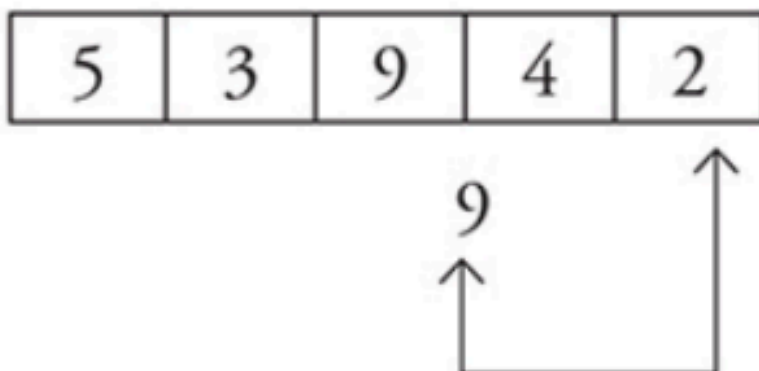
largest = 5 after first comparison



largest = 9 after second comparison



largest = 9 after third comparison



largest = 9 after last comparison

- Another common algorithm is calculating the sum and average of a group of numbers.
- This is fairly straightforward since it is a concept you have already learned in math class.



- One key element to remember is that you need to keep count of how many numbers you have added together so you will be able to calculate the average.

## Lists

- **Lists:** are a collection of items, such as a grocery list or a playlist of music.
- A list in a program can be a collection of numbers, words, phrases, or a combination of these.
- Usually a list only contains one type of data in a single list, but some programming languages do allow different types of data in the same list.
- Lists provide the ability to store more than one value in the same variable, separated by commas, when the variable is defined as a list.
- Lists are also called **arrays** in some programming languages.

Text

```
listName ← [ ]
```

Block

```
listName ← 
```

Text

```
listName ← [value1, value2, value3, . . . ]
```

Block

```
listName ← 
```

Text

```
list2 ← list1
```

Block

```
list2 ← list1
```

## List Indices

- Individual items in a list are called **elements** and are accessed by position using an index.
- **Index positions:** are always integers and are enclosed within square brackets [index].

Text :

```
listName[i]
```

Block:

```
listName [i]
```

## Built-in Methods

- Most programming languages will have built-in procedures, or “methods,” of common functionality to use with lists.
- Adding an item to a specific position in a list.
- The INSERT command causes elements to the right of the indicated index position, i, to shift right one position to make room for the new element.

Text :

```
listName[i] ← listName[j]
```

Block:

```
listName [i] ← listName [j]
```

- Appending an item to the end of the list.
- The APPEND command will add the new element to the end of the list, so no index position is needed.
- The size of the list increases by one.

Text :

```
APPEND (listName, value)
```

Block:

```
APPEND listName, value
```

- Removing an item from a list.
- The REMOVE command deletes the element at the provided index position and shifts the remaining elements one position to the left.
- The size of the list decreases by one.

Text :

```
REMOVE (listName, i)
```

Block:

```
REMOVE listName, i
```

- **Length:** The length of a list is the number of elements in the list.

Text :

```
LENGTH (listName)
```

Block:

```
LENGTH listName
```

#### Checking Each Item in a List

```
FOR EACH item IN list
```

- The above statement is a loop that will automatically repeat the code for each element in the list.
  - This is called **traversing** a list.
- The programmer chooses the name for the iteration variable "item".

- Each pass of the loop will assign the value of the next element in the list to the variable "item".
- Processing lists with a FOR EACH loop takes advantage of features of both structures.

Text:

```
FOR EACH item IN listName
{
    <block of statements>
}
```

Block:

```
FOR EACH item IN listName
    block of statements
```

## Common Algorithms

- There are algorithms that are frequently needed for processing lists with iteration.
- These include finding the largest or smallest number in a list.
- Here is an example using a REPEAT UNTIL loop with a list of grades.

<code>index ← 2</code>	This variable represents the index position for a list element.
<code>largest ← gradeList[1]</code>	This sets the largest number to be the first list element.
<code>REPEAT UNTIL (LENGTH(gradeList) = index)</code>	This loop will process each element in the list. Our condition checks to see if our index variable equals the length of the list.
<code>{</code>	
<code>IF (largest &lt; gradeList[index])</code>	This checks each list element to see if it is larger than the current value of the variable <i>largest</i> .
<code>{</code>	
<code>largest ← gradeList[index]</code>	This sets a new value for the variable <i>largest</i> .
<code>}</code>	
<code>index ← index + 1</code>	This updates the index by 1.
<code>}</code>	

- Another common algorithm is finding the sum and average of the values in a list.

```
sum ← 0
FOR EACH grade IN gradeList
    sum ← sum + grade
DISPLAY "The sum is: ", sum
DISPLAY "The average is: ", sum/LENGTH gradeList
```

## Searching

- **Searching:** deals with finding the needed element from everything in the dataset or determining that it is not there.
- **Linear Search:** Linear searches, also called sequential searches, check each individual record, starting at the beginning and going to the end, one after the other in order to either find the desired data or to determine it is not in the dataset.

Search for 10

↓	↓	↓	↓	↓	↓	↓	↓	↓	
8	65	42	29	14	5	2	15	10	9

FOR EACH num in list

```

{
    IF (num = 10)
    {
        DISPLAY ("Found 10!")
    }
}

```

- **Binary Search:** Binary searches are far more efficient than linear searches.
  - However, data must be sorted to use a binary search.
  - The binary search is considered a "divide and conquer" algorithm because it divides the dataset into two equal parts.

## Procedures

- **Procedures:** are also called functions in some programming languages.
- These are sections of code that will be executed only when they are called by the main program or another procedure.
- They must be defined in a program before they can be used in the program.

Text: The code for the procedure goes between the braces {}.

```

PROCEDURE procedureName(parameter1, parameter2, . . . )
{
    <block of statements>
}

```

Block:

```

PROCEDURE procedureName parameter1, parameter2, . . .
{
    block of statements
}

```

## Parameters/Arguments

- The use of parameters can make procedures more flexible.
- **Parameters:** allow the calling program to send values to the procedure.
- They are passed to the procedure as arguments when the procedure is called.
- The values sent to the procedure can be different each time, making the procedure more flexible through the ability for multiple calls to the same section of code.



## Calling a Procedure

- When the procedure is "called," the program will pause at that location and execute the code in the procedure.
- When the procedure finishes, control returns back to the line of code where the call occurred.
- This is another programming structure that modifies the sequential flow of the program.
- **Procedural abstraction:** You only need to know the name of the procedure, the number and type of parameters, and the output to expect.

## Random Values

- **Random number generator programs:** are useful tools for writing software, mainly in designing games.
- A random number generator picks a number at random out of a range of values.

Text: RANDOM (a, b) Block: RANDOM <span style="border: 1px solid black; padding: 2px;">a, b</span>	Generates and returns a random integer from a to b, including a and b. Each result is equally likely to occur. For example, RANDOM (1, 3) could return 1, 2, or 3.
---	---

## Return

- **Procedures:** have an optional feature called a return statement.
- The return statement has two uses.
- One purpose is to end a procedure before the end of the code is reached.
- No other code in the procedure will be executed after the return statement.
- The other use is to send a value back to the calling program.

Text :

RETURN (expression)

Block :

RETURN expression

- **Built-in Procedures:** Built-in procedures are prewritten and tested code that are included with the programming language.
- **DISPLAY():** DISPLAY() is a built-in procedure used for this course on the exam.
  - We do not know how this procedure is coded, only that we can use it multiple times, pass it different types and values to print, and that it works.

Text :

DISPLAY (expression)

Block :

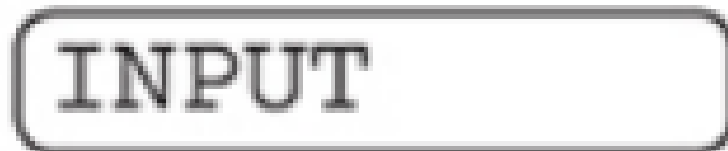


- **INPUT():** It accepts data from the user, usually from the keyboard.
  - When the programming language sees this command, it will pause the program and wait for something to be typed on the keyboard.

Text :

INPUT ( )

Block :



## APIs AND "Libraries"

- For each programming language, there are prewritten programs to provide commonly needed functionality, and these programs are stored in libraries, which are folders with several programs.
- API stands for "Application Programming Interface."
- The API documentation provides the information needed to set up the interface and use the newly connected software.

## Random

- Generating random numbers is a frequently needed feature in programs.
- Most programming languages have a library of prewritten code for a variety of random number generators.

- RANDOM needs two values passed to it using arguments, the beginning and ending range for the selected random number.
- **Simulations:** Simulations are designed to represent and mirror the real world for testing.

## Analyzing Algorithms

- An *instance* of a problem is a specific example.
- A *decision* problem has a yes or no answer.
- An *optimization problem* is one that should find the best solution for the problem.

## Algorithm Efficiency

- The efficiency of algorithms deals with resources needed to run it in terms of how long it will take and how much memory will be needed.
- This becomes especially important with extremely large datasets, and efficiency is usually stated in terms of the size of the input.
- **Efficiency:** can be determined by mathematically proving it and informally measured by actually running it on datasets of different sizes and measuring how long it took and the memory resources needed.

## Limits of Algorithms

- Algorithms have limits, and there are some problems for which we do not have efficient enough algorithms to solve.
- These algorithms can't run in a reasonable amount of time with our current technology.
- **Heuristic approach:** This is an approach that may not be optimal or the best but is close enough to use as a solution.

## Street Traffic Models

- Simulation, as a tool, is used when studying traffic systems when the system is too complicated.
- The advantage of simulation tools is that they provide visual demonstrations for different scenarios, both present and future.
- These models are necessary tools for planning and operating traffic systems, as they help to predict the behavior of vehicles in the traffic system.

## Solar Activity Models

- All engineering disciplines use simulation tools that help them study phenomena specific to their field of expertise.
- **Simulations** are now part of the design process.
- **Models** require considerable time and effort to develop.
- Simulation models are used to study the solar system.

- **Modeling and simulation** is a powerful method to evaluate the design of a space system.
- **Simulation models** represent valuable knowledge that scientists use to build systems to explore space.

## Undecidable/Unsolvable Problems

- **Decidable problem:** is one where an algorithm can be written that results in a correct "yes" or "no" answer for all inputs.
- Determining if a number is prime is an example of a decidable problem.
- **Undecidable problem:** does not have an algorithm that can give a correct "yes" or "no" for all cases of the problem.

## Chapter 4: Computer Systems and Networks

### The Internet

- **Internet:** is a network of networks.
- The word Internet came from "interconnection of computer networks."
- The Internet is very hardware driven with wires, cables, and devices such as routers and servers.
- *Routers* are computing devices along a path that send the information along to the next stop on the path.
- **Routing:** is the process of finding a path from sender to receiver
- **Bandwidth:** is a measure of the maximum amount of data that can be transferred through a channel or network connection.
  - It's measured in bits per second, and it determines how quickly you can download and upload files from the internet.
- **Internet protocol (IP):** is responsible for addressing and routing your online requests.
- **Transmission control protocol (TCP):** is a protocol that defines how computers send packets of data to each other.
- **User datagram protocol (UDP):** is a protocol that allows computer applications to send messages without checking for missing packets to save on time needed to retransmit missing packets.

### Scalability

- **Scalability:** is the ability for a system, network or process to handle a growing amount of work in an efficient manner.
- It can also be defined as the capacity to increase services and products quickly with minimal interruption and cost.
  - This is especially important in software engineering, where scalability is crucial when developing applications that are expected to handle large amounts of traffic or data.

The diagram illustrates a network topology with two alternate paths between a central diamond-shaped core and four end devices. The components and their connections are as follows:

- Top Path:** An "End Computer" is connected to a "Switch", which is connected to a "Router".
- Bottom Path:** A "Router" is connected to a "Switch", which is connected to a "Computer" labeled "Start".
- Left Path:** A "Computer" is connected to a "Switch", which is connected to a "Router".
- Right Path:** A "Router" is connected to a "Switch", which is connected to a "Computer".
- Central Core:** The four "Router" nodes are interconnected in a diamond shape. The top and bottom routers are connected to the left and right routers. The connections between the top and bottom routers, and between the left and right routers, are labeled "Alternate".

- **Hardware failure:** is when a hardware device, such as a computer or printer, stops working properly due to an issue with the physical components.
- The cause of the failure can be anything from electrical wiring issues to incorrect installation and configuration of hardware components.
- In any case, diagnosing and repairing hardware failures can be difficult without proper tools and experience in dealing with this type of issue.

- **Operational failures:** are any issues or breakdowns in the operation of a business, machine, system, process, etc.
- They can range from unexpected downtime to incorrect results due to faulty programming.
- Operational failures can have significant impacts on profitability and reputation if not addressed swiftly and appropriately.

- The internet has cables and wires spanning the world that connect computers.



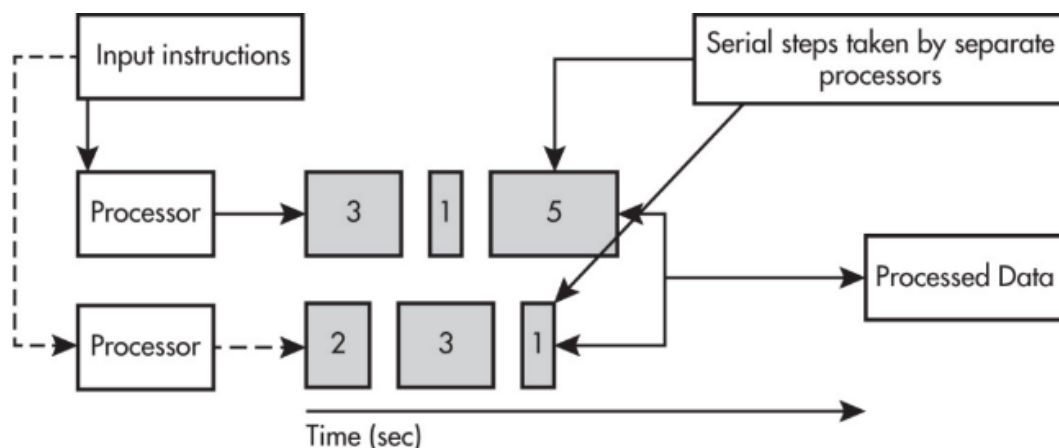
- **Natural disasters:** could cause the hardware to be destroyed, bringing the network activity to a halt.
- **Solar Flare:** is an intense radiation that is released from the sun.
  - This happens because of the released from the sun.

## Cyberattacks

- **Cyberattacks:** are malicious attempts to damage or disrupt computer systems, networks, and data.
- They can be carried out by individuals, groups, or organizations with malicious intent.
- Cyberattacks typically involve the use of malware such as viruses and ransomware that allow attackers to gain access to a system for a variety of nefarious purposes including stealing data and financial information or launching denial-of-service attacks.
- Some cyberattack methods used today include phishing campaigns; social engineering attacks; website defacement; distributed denial of service (DDoS) attacks; SQL injection exploits; man-in-the middle (MITM) attack vectors etc.

## Parallel and Distributed Computing

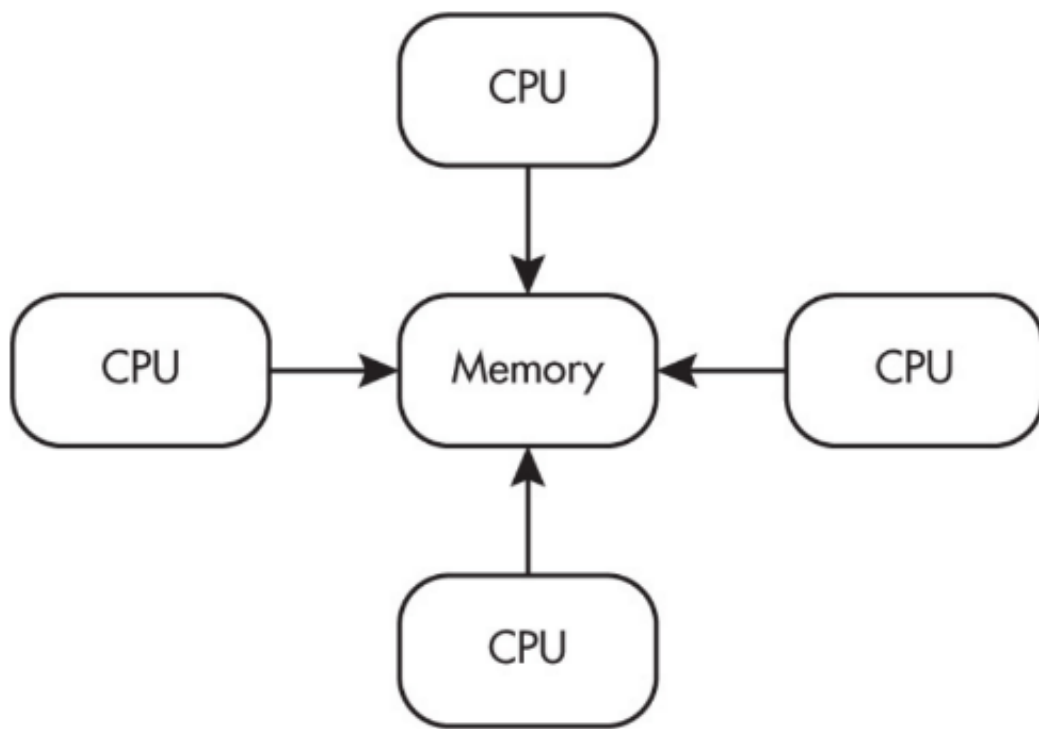
### Parallel Computing



- **Parallel computing solution** takes as long as the longest of the tasks done in parallel.
  - A parallel computing solution takes as long as its sequential tasks plus the longest of its parallel tasks.
- **Parallel computing** can consist of a parallel portion and a sequential portion.

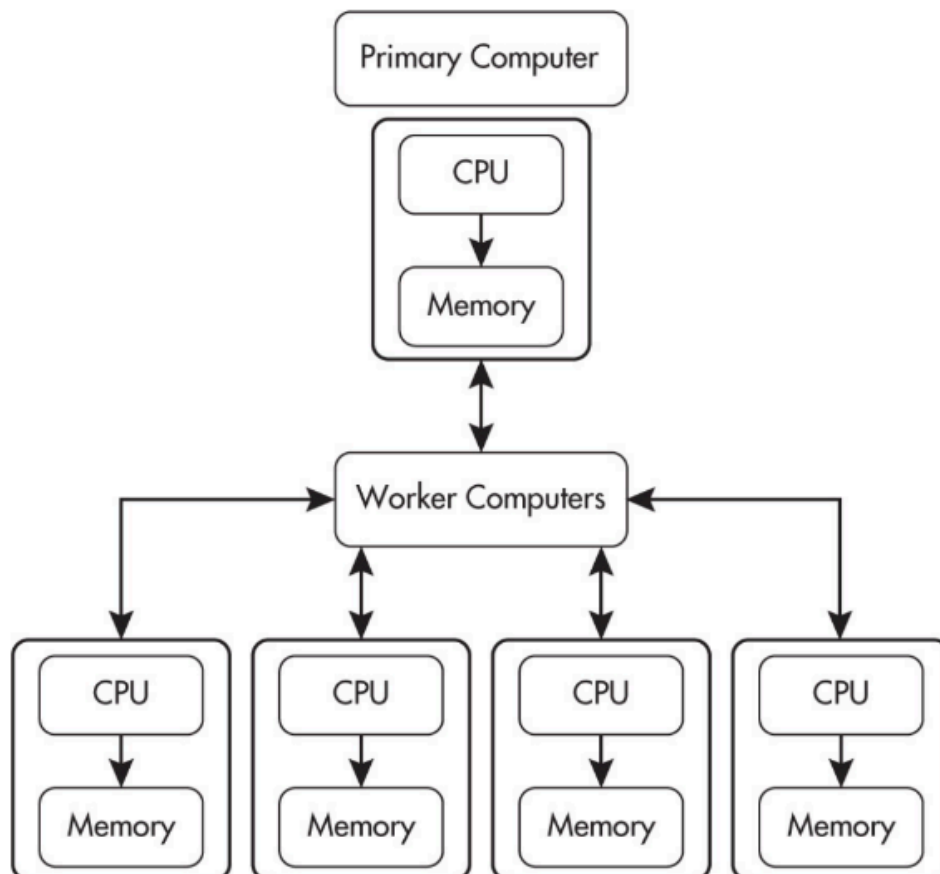
### Why Is Parallel Computing Used?

- **Parallel computing** is needed for real-world simulations and modeling.
- **Multiple processors** can operate independently but share the same memory resources.



## Distributed Computing

- **Distributed computing** allows problems to be solved that could not be solved on a single computer because of either the processing time or storage needs involved.
- Parallel computing uses a single computer with multiple processors.
- Distributed computing uses multiple computing devices to process those tasks.



## Chapter 5: Impact of Computing

# New Ways to Communicate and Interact

- Thanks to the Internet and the ease of collaboration and sharing, programs and apps (software applications) can be quickly and easily shared with people worldwide.
- These can sometimes have a huge impact, positive or negative, on people, and sometimes have additional results not originally foreseen by the software developers.
- **World Wide Web:** which was originally designed for scientists to share their research.
- **Targeted advertising:** which can be helpful for businesses and consumers when looking for a specific item.
- **Social media:** which has been used to stream events across the globe, sometimes helps to change history.
- **Machine learning and data mining:** help find patterns and identify insights in data, leading to new innovations.
- **Online learning:** is an education model that would not be possible without the tools of communication available via the Internet.
- **Programmers and businesses:** try to identify potential negative uses, but it is seldom possible to think of all the ways other people could use an innovation.

## Access to Information

### Cloud Computing

- **Cloud computing:** offers new ways for people to communicate, making collaboration easier and more efficient.
- Storing documents in the "cloud" simply means they are stored on a computer server at a location different than where the owner of the files is located.

### Digital Divide

- **Technology** has had a major impact on the world, enabling innovation through the sharing of resources and computational artifacts.
- It also allows us to virtually meet with people from anywhere.
- It is helping us on the path of becoming a true global society.
- The impact of the digital divide includes access to information, knowledge, markets, and different cultures.

### Bias in Computing Innovations

- **Bias:** which is intentional or unintentional prejudice for or against certain groups of people, shows up in computing innovations too.
- **Humans:** write the algorithms, and our biases can make their way into the algorithms and the data used by innovations without us realizing it.
- **Artificial intelligence programs:** are used more and more in ways such as screening applications of job candidates, determining if a person merits credit to purchase a house, and locating what areas have more crime.

## Crowdsourcing

- **Crowdsourcing:** allows people to share information and ask the “crowd”— anyone who accesses the site—for feedback, to help solve problems, find employment, or for funding.
  - Another use of crowdsourcing is when scientists share data and ask nonscientists, or “citizen scientists,” to look for and report on patterns or other interesting features of the data or to “donate” computer time during periods of time their machine is inactive.
  - This helps to “scale up” processing capability at little to no cost to the organization seeking the resources.

## Legal and Ethical Concerns

- Anything a person creates, including any computational artifacts created with a computer, is the intellectual property of that person.
- Material created by someone else that you use in any way should always be cited.
- **Peer-to-peer networks** exist that are used to illegally share files of all types.
- Devices that continually monitor and collect data, such as a voice-activated device we install or video cameras used for facial recognition posted in our communities, can have legal and/or ethical issues.

## Creative Commons

- **Creative Commons:** provides a way for creators of software, images, music, videos, and any computational artifact to share their creations with stipulations for sharing and permission from the author clearly indicated.
- **Digital data:** is easy to find, copy, and paste, so ensuring you have written permission from the creator or owner is important.

## Open-Source Software

- Not only do we have access to data but to software as well.
- **Open source** software: is software that is freely shared, updated, and supported by anyone who wants to do so.
- The availability of this software for everyone has greatly expanded people’s abilities to participate in a variety of tasks that many would not have been able to participate in otherwise.

## Open Access

- The sharing of huge amounts of public data by organizations, such as the U.S. government, provides the opportunity for anyone to search for information or to help solve problems.
- In addition, the availability of open databases in a variety of fields—including science, entertainment, sports, and business—has benefited people everywhere.

## Search Trends and Analytics

- **Social media sites:** as well as search engines publish what the most frequent searches and posts are about.
- Our browsers keep a list of our most frequently visited sites on their home page to help us out.
- The search engines are able to identify when more people than usual are watching a video or searching for a topic.
- **Analytics:** identify trends for marketing purposes and help businesses determine what and where customers are searching for their products and their competitors' products, how long an item sits in a virtual shopping cart, and when people buy.

## Data Mining

- **Data mining:** is a field of study that analyzes large datasets.
- **Machine learning:** is a subset of data mining.
- Machine learning uses algorithms to analyze data and predict behavior and is used in Artificial Intelligence (AI).

## Personally Identifiable Information (PII)

- Any information that identifies you is considered Personally Identifiable Information (PII).
- It includes data such as your address, age, or social security number.
- It also includes data about you, such as your medical or financial information.
- Our PII information is also used by websites to show us certain information or related topics based on our prior visits.

## Privacy

- **Digital footprints and fingerprints:** are the trail of little pieces of data we leave behind as a sign of our presence as we go through our daily lives.
- Many people willingly provide personal information to sites to gain access or privileges, whether it's through sports teams, shopping, or restaurants.
  - Their data is stored and may be sold with or without their knowledge or permission.
- Many web browsers now have "incognito" or "private" modes so that web searches and file downloads are not recorded in the web history.
- Some web browsers attest that they do not track and retain your search data.

## Protecting Our Data

- Many aspects of our lives are much easier today because of the easy access to all sorts of sites and information that the Internet provides.
- This can range from shopping, entertainment, and sports sites to price comparisons.



- **Cybersecurity:** has a global impact because now anyone from anywhere can attempt to gain unauthorized entry to someone else's computer, data, servers, or network.

## Security

- The security of our data deals with the ability to prevent unauthorized individuals from gaining access to it and preventing those who can view our data from changing it.
- **Strong passwords:** help block those trying to gain unauthorized access.
- **Multifactor authentication:** is another layer that is increasingly used.

## Cybersecurity

- **Cybersecurity:** protects our electronic devices and networks from attacks and unauthorized use.
  - These attacks can come in many forms and can have a major impact on those affected.
- Different types of attacks cause different problems.
- **Data:** may be damaged or the device may be used to further spread the malware.
- **Phishing:** attacks create e-mail and/or websites that look a legitimate hoping to induce a person to click on the malicious link.
- **Computer viruses:** are like human viruses.
  - They attach themselves to, or are part of, an infected file.
- **Keylogging software:** is a form of malware that captures every keystroke and transmits it to whomever planted it.

## Cryptography

- **Cryptography:** is the writing of secret codes.
- **Encryption:** is converting a message to a coded format.
- **Deciphering:** the encrypted message is called decryption.
- **Security:** also relates to encrypting data before it is transmitted to ensure it remains secure if it is intercepted during transmission.

## Public Key Encryption

- **Public key encryption:** uses open standards, meaning the algorithms used are published and available to everyone and are discussed by experts and interested parties and known by all.
  - The key is what keeps information secret until the person it is intended for decrypts it.

## Securing the Internet

- The **Internet** is based on a "trust" model.

- This means that digital certificates can be purchased from Certificate Authorities (CAs), which identify trusted sites.
  - They issue certificates that businesses, organizations, and individuals load to their websites.
- The certificates verify to web browsers that the encryption keys belong to the business, thereby enabling online purchases and the sending and receiving of secure documents.