1. **Computer Algorithms and Problem Solving**  
   CST111—Introduction to Information Technology

3. **Problem Solving**
   - Problem solving—the act of finding a solution to a perplexing, distressing, vexing, or unsettled question
     - *How do you define problem solving?*

4. **Problem Solving**
   - *How to Solve It: A New Aspect of Mathematical Method* by George Polya
   - “How to solve it list” written within the context of mathematical problems
   - But the list is quite general

5. **Problem Solving**
   - *How do you solve problems?*
     - Understand the problem
     - Devise a plan
     - Carry out the plan
     - Look back

6. **Strategies**
   - Ask questions!
     - What do I know about the problem?
     - What is the information that I have to process in order to find the solution?
     - What does the solution look like?
     - What sort of special cases exist?
     - How will I recognize that I have found the solution?

7. **Strategies**
   - Ask questions! Never reinvent the wheel!
   - Similar problems come up again and again in different guises
   - A good programmer recognizes a task or subtask that has been solved before and plugs in the solution
   - *Can you think of two similar problems?*

8. **Strategies**
   - Divide and Conquer!
   - Break up a large problem into smaller units and solve each smaller problem
     - Applies the concept of abstraction
     - The divide-and-conquer approach can be applied over and over again until each subtask is manageable

9. **Computer Problem-Solving**
   - Analysis and Specification Phase
     - Analyze and specification
   - Algorithm Development Phase
     - Develop algorithm and test algorithm
   - Implementation Phase
     - Code algorithm and test algorithm
   - Maintenance Phase
     - Use and maintain

11. **Algorithms**
    - Algorithm—a set of *unambiguous* instructions for solving a problem or sub-problem in a *finite* amount of *time* using a *finite* amount of *data*
      - Abstract Step—an algorithmic step containing unspecified details
      - Concrete Step—an algorithm step in which all details are specified

12. **Developing an Algorithm**
Two methodologies used to develop computer solutions to a problem:
- Top-down design focuses on the tasks to be done
- Object-oriented design focuses on the data involved in the solution (discussed in Chapter 9)

13 Top-Down Design
- Process continues for as many levels as it takes to make every step concrete
- Name of (sub)problem at one level becomes a module at next lower level

14 Summary of Methodology
- Analyze the Problem
  - Understand the problem!!!
  - Develop a plan of attack
- List the Main Tasks (becomes Main Module)
  - Restate problem as a list of tasks (modules)
  - Give each task a name
- Write the Remaining Modules
  - Restate each abstract module as a list of tasks
  - Give each task a name
- Re-sequence and Revise as Necessary
  - Process ends when all steps (modules) are concrete

19 Assignment
- The process of storing a value into RAM memory
- A variable is a programmer-defined identifier (word) use is a program that represents the memory location where the value is stored
- When the variable name is used in a code statement, it is a placeholder (represents) the value stored in RAM

21 Control Structures
- An instruction or algorithm that determines the order in which other instructions in a program are executed
  1. Sequence—step by step in order
  2. Selection—decisions (conditional statements)
  3. Iteration—looping (repeating statement blocks)

22 Sequence Structure
- One statement executed after another in sequence
- Executed in order stored in RAM (program order)
- chicken move forward 5 meters
- chicken roll left 1 revolution
- chicken turn left ¼ turn
- chicken think “Amazing!”

23 Selection (Decision) Structure
- Requires a Boolean test
  - The test result is either true or false
  - One of two paths are taken within the code
- If (wantCookie is true) Then
  - cookie move to camera
  - cookie move up 10 meters
  - Else
  - cookie move to camera
  - End If

25 Iteration (Loop/Repetition) Structure
- One or more statements are executed repeatedly
- The loop continues while the controlling condition is true
While (ballHeight < kelly’s height)
        kickTheBall
        add 0.2 to ballHeight
    Loop

31 **Composite Data Types**
- Records—a named heterogeneous collection of items in which individual items are accessed by name:
  - I.e. we could bundle name, age and hourly wage items into a record named Employee
- Arrays—a named homogeneous collection of items in which an individual item is accessed by its position (index) within the collection

32 **Composite Data Types**
Employee
    name
    age
    hourly/Wage
Following algorithm, stores values into the fields of record:

Employee employee  // Declare and Employee variable
Set employee.name to “Frank Jones”
Set employee.age to 32
Set employee.hourlyWage to 27.50

34 **Arrays**
- As data is being read into an array, a counter is updated so that we always know how many data items were stored
  - If the array is called list, that with which we are working is from list[0] to list[length-1]

37 **Sequential Search of an Unsorted Array**
- A sequential search examines each item in turn and compares it to the one we are searching.
  - If it matches, we have found the item. If not, we look at the next item in the array.
  - We stop either when we have found the item or when we have looked at all the items and not found a match
- Thus a loop with two ending conditions

38 **Sequential Search Algorithm**
Set Position to 0
Set found to FALSE
WHILE (position < length AND NOT found )
    IF (numbers [position] equals searchitem)
        Set Found to TRUE
    ELSE
        Set position to position + 1

39 **Booleans**
- Boolean operators—a Boolean variable is a location in memory that can contain either true or false
  - Boolean operator AND returns TRUE if both operands are true and FALSE otherwise
  - Boolean operator OR returns TRUE if either operand is true and FALSE otherwise
  - Boolean operator NOT returns TRUE if its operand is false and FALSE if its operand is true

40 **Sorted Arrays**
- The values stored in an array have unique keys of a type for which the relational
operators are defined.
- Sorting rearranges the elements into either ascending or descending order within the array
- A sorted array is one in which the elements are in order

### 41 Sequential Search in a Sorted Array
- If items in an array are sorted, we can stop looking when we pass the place where the item would be if it were present in the array

### 45 Binary Search
Sequential search—search begins at the beginning of the list and continues until the item is found or the entire list has been searched
Binary search (list must be sorted)—search begins at the middle and finds the item or eliminates half of the unexamined items; process is repeated on the half where the item might be

### 49 Sorting
- Sorting—arranging items in a collection so that there is an ordering on one (or more) of the fields in the items
- Sort Key—the field (or fields) on which the ordering is based
- Sorting algorithms—algorithms that order the items in the collection based on the sort key

### 50 Selection Sort
- Given a list of names, put them in alphabetical order
  - Find the name that comes first in the alphabet, and write it on a second sheet of paper
  - Cross out the name off the original list
  - Continue this cycle until all the names on the original list have been crossed out and written onto the second list, at which point the second list contains the same items but in sorted order

### 51 Selection Sort
- A slight adjustment to this manual approach does away with the need to duplicate space
  - As you cross a name off the original list, a free space opens up
  - Instead of writing the value found on a second list, exchange it with the value currently in the position where the crossed-off item should go

### 56 Bubble Sort
- Bubble Sort uses the same strategy:
  - Find the next item
  - Put it into its proper place
- But uses a different scheme for finding next item
- Starting with last list element, compare successive pairs of elements, swapping whenever the bottom element of the pair is smaller than the one above it

### 61 Insertion Sort
- If you have only one item in the array, it is already sorted.
- If you have two items, you can compare and swamp them if necessary, sorting the first two with respect to themselves.
- Take the third item and put it into its place relative to the first two
- Now the first three items are sorted with respect to one another

### 62 Insertion Sort
- The item being added to the sorted portion can be bubbled up as in the bubble sort

### Insertion Sort
Set current to 1
WHILE (current < length)
  Set index to current
  Set placeFound to FALSE
  WHILE (index > 0 AND NOT placeFound)
    IF (data[index] < data[index - 1])
      Swap data[index] and data[index - 1]
      Set index to index - 1
    ELSE
      Set placeFound to TRUE
    END IF
  END WHILE
  Set current to current + 1
END WHILE

Subprogram Statements
- We can give a section of code a name and use that name as a statement in another part of the program
- When the name is encountered, the processing in the other part of the program halts while the named code is executed

Subprogram Statements
- What if the subprogram needs data from the calling unit?
- Parameters—identifiers listed in parentheses beside the subprogram declaration; sometimes called formal parameters
- Arguments—identifiers listed in parentheses on the subprogram call; sometimes called actual parameters

Recursion
- Recursion—ability of a subprogram to call itself
- Base case—the case to which we have an answer
- General case—the case that expresses the solution in terms of a call to itself with a smaller version of the problem

Recursion
- For example, the factorial of a number is defined as the number times the product of all the numbers between itself and 0:
  \[ N! = N \times (N - 1)! \]
- Base case
  Factorial(0) = 1 (0! is 1)
- General Case
  Factorial(N) = N \times Factorial(N-1)

Recursion

Write “Enter n”
Read n
Set result to Factorial(n)
Write result + “ is the factorial of “ + n

Factorial(n)
IF (n equals 0)
  RETURN 1
ELSE
  RETURN n \times Factorial(n-1)

Quicksort algorithm
- With each attempt to sort the stack of data elements, the stack is divided at a splitting value, splitVal, and the same approach is used to sort each of the smaller stacks (a
• Process continues until the small stacks do not need to be divided further (the base case)
• The variables first and last in Quicksort algorithm reflect the part of the array data that is currently being processed

**Important Threads**
• Information Hiding—the practice of hiding the details of a module with the goal of controlling access to it
• Abstraction—a model of a complex system that includes only the details essential to the viewer
• Information Hiding and Abstraction are two sides of the same coin

**Important Threads**
• Data abstraction—separation of the logical view of data from their implementation
• Procedural abstraction—separation of the logical view of actions from their implementation
• Control abstraction—separation of the logical view of a control structure from its implementation

**Important Threads**
• Identifiers—names given to data and actions by which:
  • We access the data and
    • Read firstName, Set count to count + 1
  • Execute the actions
    • Split(splitVal)
• Giving names to data and actions is a form of abstraction