algorithm: The logical steps necessary to solve a problem in a computer; a function or a part of a function.

application software: Computer software developed to support a specific user requirement. Contrast with system software.

application-specific software: Software that can be used only for its intended purpose.

arithmetic-logical unit (ALU): A component of the central processing unit.

assembler: System software that converts a source program into executable object code.

assembly language: A programming language in which there is a one-for-one correspondence between the symbolic instruction set of the language and the computer's machine language.

central processing unit (CPU): The part of a computer that contains the control components; that is, the part that interprets instructions. In a personal computer, a microchip containing a control unit and an arithmetic logical unit.

code errors: A syntactical error in a program; it generates an error or warning.

compiler: System software that converts a source program into executable object code; traditionally associated with high-level languages.

computer hardware: The physical equipment that comprises the computer.

computer language: Any of the syntactical languages used to write programs for computers, such as machine language, assembly language, C++, COBOL, and FORTRAN

computer software: The system and application software required to accomplish a task.

computer system: The set of computer components required for a complete system, consisting of at least an input device, an output device, a monitor, and a central processing unit.

data item: An elementary component a data with a computer.

executable file: A file that contains program code in its executable form; the result of linking the source code object module with any required library modules.

executable program: An error-free program that can be run on a computer.

function: A named unit of computation. It executes a block of code.

functional paradigm: A language paradigm that maps a list of inputs to a list of outputs.

general-purpose software: Software that can be used for more than one application.

hardware: Any of the physical components of a computer system, such as the keyboard or a printer.

high-level language: A (portable) programming language designed to allow the programmer to concentrate on the application rather than the structure of a particular computer or operating system.

imperative paradigm: programming paradigm that uses statements that change a program's state.

linker: The program by which an object module is joined with precompiled functions to form an executable program.

input system: Devices, such as a keyboard and a mouse, used to enter data into a computer.

loader: The operating system function that fetches an executable program into memory for running.

logic errors: An error in the design of a function or program that causes it to produce invalid output.

logic paradigm: A set of facts and a set of rules, both of which are used in answering queries.

machine language: The instructions that are native to the central processor of a computer and are executable without assembly or compilation.

object-oriented paradigm: The set of procedures that can be applies to a particular type of data package and must be packaged with the data.

operating system: The software that controls the computing environment and provides a user interface.

output system: Devices, such as a monitor or printer, where output is displayed or printed.

primary memory: The computer component in which programs and data are stored temporarily during processing.

procedural paradigm: Derived from imperative programming, contains a series of computational steps to be carried out.

procedure: Synonym for a function

program design: A two-step process that requires understanding the problem and then developing a solution.

program errors: Errors resulting from bad program (specification errors, code errors, logic errors).

program testing: The iterative process of executing a program to verify that it runs correctly and produces valid output.

regression testing: The process of testing a program after changes have been made.

secondary storage: Synonym for auxiliary storage.

software: The application and system programs necessary for computer hardware to accomplish a task, including their documentation and any required procedures.

source file: A file that contains a C++ program.

specification errors: A problem definition that is either incorrectly stated or misinterpreted.

symbolic language: A computer language, one level removed from machine language, in which there is a mnemonic identifier for each machine instruction and which has the capability of using symbolic data names.

system development software: A software used to define, test, and implement ne software application or program.

system software: Any software whose primary purpose is to support the operation of the computing environment.

system support software: Software used for non–application processing, such as system utilities.

text editor: Software that maintains text files, such as a word processor or a source program editor.

Unified Modeling Language (UML): A standard tool for designing, specifying, and documenting many aspects of a computing system.

Computer systems are made up of two major components:

1. hardware (CPU, memory, secondary storage, output system, and communication system)
2. software (system software and application software).

Computer languages are used to develop software. The computers themselves run in machine language.

Over the years programming languages have progressed through symbolic languages to the many high-level languages used today.

Language paradigms (procedural, object-oriented, functional, logic) describe the approach used to solve problems on the computer. C++ is based on the procedural and object-oriented paradigms.

Program design is a two-step process that requires understanding the problem and then developing a solution.

Algorithms have two important characteristics; they are independent of the computer system and they accept data as input and process data into an output.

Program development turns the program design into a computer system in four steps:

1. write the program,
2. compile it,
3. link it,
4. execute it.

Testing a program requires that every instruction and every possible situation is validated.

CP-1. List six parts of computer hardware.

1. CPU,
2. main memory,
3. secondary storage,
4. input system,
5. output system
6. communication system

CP-2. List the components of the CPU.

1. arithmetic-logical unit (ALU)
2. control unit
3. set of registers

CP-3. List some input and output devices.

Input: keyboard, mouse, pen, stylus, touch screen, audio input

Output: monitor, printer

CP-4. Differentiate between primary memory and secondary storage.

Primary memory is where programs and data are stored temporarily during processing. The contents of primary memory are lost when we turn off the computer. Is used to: to store the operating system, to store the program, and to store data.

Programs and data are stored permanently in secondary storage (hard disk, CD/DVD, flash drives).

CP-5. Describe the two categories of computer software.

System Software (manages the computer resources):

- *Operating System:* provides services such as a user interface, file and database access, and interfaces to communication systems
- *System Support:* provides system utilities and other operating services; provides performance statistics for the operational staff and security monitors to protect the system and data
- *System Development:* includes the language translators that convert programs into machine language for execution, debugging tools to assure that the programs are error-free

Application Software (responsible for helping users solve their problems):

- *General-Purpose Software:* is purchased from a software developer and can be used for more than one application.
- *Application-Specific Software :* can be used only for its intended purpose

CP-6. Differentiate between machine, symbolic, and high-level languages.

The only language understood by a computer is its machine language.

Symbolic language uses mnemonic symbols to represent machine language instructions. Each machine instruction had to be individually coded. Required programmers to concentrate on the hardware they were using.

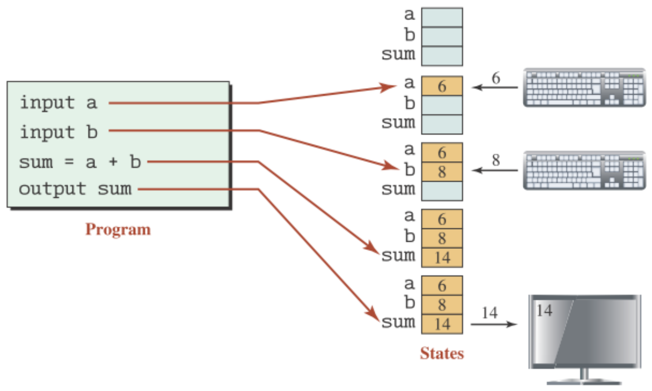
High-level languages are portable to many different computers. Are designed to relieve the programmer from the details of the assembly language but must be converted to machine language (compilation process)

CP-7. To which category of languages does the C++ belong?

High-level Language

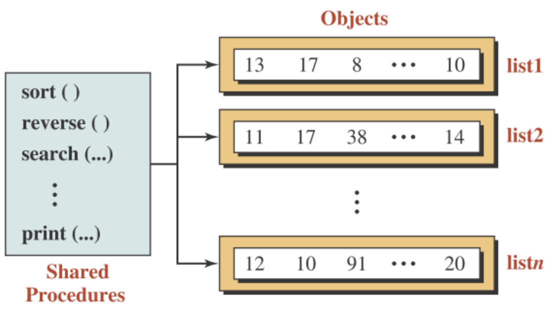
CP-8. Briefly describe the procedural paradigm.

The execution of each command changes the state of the memory related to that problem.



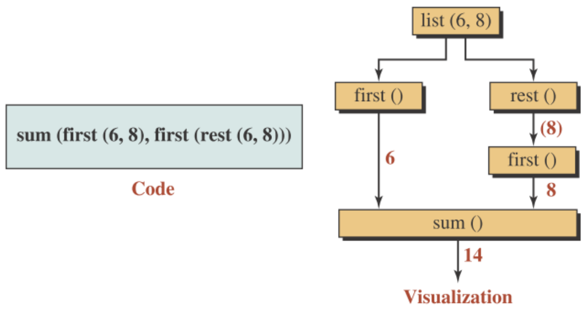
CP-9. Briefly describe the object-oriented paradigm.

Defines that the set of procedures that can be applied to a particular type of data package needs to be packaged with the data.



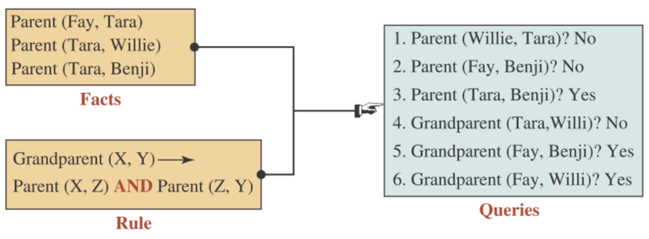
CP-10. Briefly describe the functional paradigm.

Is concerned with the result of a mathematical function, maps a list of inputs to a list of outputs. We are not using commands and we are not following the memory state.



CP-11. Briefly describe the logic paradigm.

Uses a set of facts and a set of rules to answer queries.


Parent (Fay, Tara)
Parent (Tara, Willie)
Parent (Tara, Benji)
Facts
Grandparent (X, Y)
Parent (X, Z) AND Parent (Z, Y)
Rule
1. Parent (Willie, Tara)? No
2. Parent (Fay, Benji)? No
3. Parent (Tara, Benji)? Yes
4. Grandparent (Tara,Willi)? No
5. Grandparent (Fay, Benji)? Yes
6. Grandparent (Fay, Willi)? Yes
Queries

CP-12. To which paradigm category does the C++ language belong?

Procedural Paradigm, and Object-Oriented Paradigm

CP-13. What is the first step in program design?

Understanding the problem:

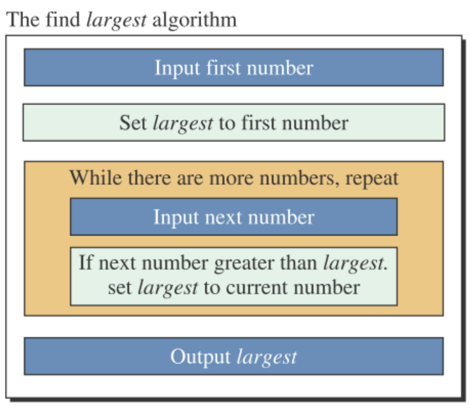
- What type of numbers are we dealing with (with fractions or without fractions)?
- Are the numbers arranged in any special sequence, such as lowest to highest?
- How many numbers can we expect?

CP-14. What is the second step in program design?

Developing a solution (creating an algorithm)

CP-15. What is algorithm generalization?

Presenting algorithm in a block diagram and generalizing all statements. Instructions must be clear for any data input,


The find *largest* algorithm
Input first number
Set *largest* to first number
While there are more numbers, repeat
Input next number
If next number greater than *largest*. set *largest* to current number
Output *largest*

CP-16. What is UML and what is its role in defining an algorithm?

The Unified Modeling Language (UML) is a standard tool for designing, specifying, and documenting many aspects of a computing system.

Uses diagrams to explain how the algorithm works.

CP-17. List four steps in the program development phase.

1. Write and Edit Programs
   Using text editors to enter, change and store character data. Preparing a source file.
2. Compile Programs
   Translation of a source file into a machine language.
3. Link Programs
   Importing library functions into the source file.
4. Execute Program
   The program reads data for processing, either from the user or from a file. After the program processes the data, it prepares the output.

CP-18. What are the four areas of program testing?

1. Verify that every line of code has been executed at least once.
2. Verify that every conditional statement in the program has executed both the true and false branches, even if one of them is null.
3. For every condition that has a range, make sure the tests include the first and last items in the range, as well as items before the first and after the last.
4. If error conditions are being checked, make sure all error logic is tested.

CP-19. Differentiate between code errors and logic errors.

Code Error are easy to detect by the complier (Syntax Error).

Logic Error prevents the program to be executed or present wrong results. Hard to find.