* computer – a programmable electronic device that can store, retrieve, and process data
* Much of human behavior and thought is characterized by logical sequences.
* Programming is planning or scheduling the performance of a task or an event.
* Computer Programming is the process of planning a sequence of steps for a computer to follow
* 3 phases in a program’s life cycle
	1. Problem-solving phase
		1. Analysis and specifications – define a problem and what the solution must do
		2. General solution (algorithm) develops a logical sequence of steps that solves the problem
		3. Verify – follow the steps exactly to see if the solution solves the problem
	2. Implementation phase
		1. Concrete solution (program) – translate the algorithm into a programming language
		2. Test – have computer follow instructions, then manually check the results, and make corrections
	3. Maintenance phase
		1. Use program
		2. Maintain – modify as needed
* Algorithm is a step-by-step procedure for solving a problem in a finite amount of time
	+ - * We use algorithms every day. Such as recipes, instructions, and directions are all examples of algorithms that are not programs
			* An algorithm must terminate in a finite amount of time for all possible conditions
* Programming language is a set of rules, symbols and specific words used to construct a computer program
	+ - * Programming forces you to write very simple, exact instructions
			* Coding the algorithm is translating an algorithm into a programming language
			* Debug is determining what is wrong and modify program
			* Think first and code later (saves time later)
			* Documentation is the written text and comments that make a program easier for others to understand, use, and modify
			* Information is any knowledge that can be communicated
			* Data is information in a form a computer can use
* Binary Code – 0 and 1
* Assembler – a program that translates an assembly language program into machine code
* Compiler – a program that translates a high-level language into machine code
* Source program – a program written in a high-level programming language
* Object program – the machine language version of a source program
* Benefit of standardized high-level languages is that they allow you to write portable code that can be used on different machines
* Compilation – computer runs the compiler program
* Execution – program loaded into memory unit, replacing compiler program
* 4 basic ways of structuring statements (instructions) in most programming languages
	1. Sequence is a series of statements that are executed one after another
	2. Conditional execute different statements depending on certain conditions
	3. Repetitive (loop) repeats statements while certain conditions are met
	4. Subprogram – structure a program by breaking it into smaller units
* 6 basic computer components
	1. memory unit – internal data storage
	2. arithmetic / logic unit – perform operations and compare values
	3. control unit – controls actions so instructions executed in correct order
	4. input devices – accept data to be processed (keyboard, mouse)
	5. output devices – present results of processing (printer, screen)
	6. peripheral devices / auxiliary storage devices (secondary storage, scanner, CD, DVD, digital camera, modems, audio sound cards and speakers)
* Hardware – physical components
* Software – programs
* Operating system – a set of programs that manages all of the computer’s resources
* Ethics and Responsibilities in the Computing Profession
	+ - * Software piracy – The unauthorized copying of software for either personal use or use by others.
			* Virus – A computer program that replicates itself, often with the goal of spreading to other computers without authorization, and possibly with intent of doing harm.
			* Software engineering – The application of traditional engineering methodologies and techniques to the development of software.
* Problem-Solving Techniques
	1. ask questions
		+ - What do I have to work with – that is, what is my data?
			- What do the data items look like?
			- How much do the data items look like?
			- How much data is there?
			- How will I know when I have processed all the data?
			- What should my output look like?
			- How many times is the process going to be repeated?
			- What special error conditions might come up?
	2. look for things that are familiar
		+ - If you’ve solved the same or similar problem before, just repeat your solution.
	3. solve by analogy
		+ - broader application of the strategy of looking for things that are familiar
	4. means ends analysis
		+ - You begin by writing down what the input is and what the output should be. Then you consider the actions a computer can perform and choose a sequence of actions that can transform the data into the results.
	5. divide and conquer
		+ - break up a large problem into smaller pieces that we can solve individually
	6. building block approach
		+ - combining smaller problems in which solutions may already exist to solve a large problem
	7. merging solutions
		+ - Whenever the solutions to subproblems duplicate steps, think about merging them instead of joining them end-to-end.
	8. mental blocks: fear of starting
		+ - rewrite the problem in your own words is a good way to focus on the subparts of the problem
	9. algorithmic problem-solving
		+ - step-by-step procedure to make the computer transform, manipulate, calculate, or process the input data to produce the desired output

Questions:

1. What is a computer program?
2. What are the three phases in a program’s life cycle?
3. Is an algorithm the same as a program?
4. What is a programming language?
5. What are the advantages of using a high-level programming language?
6. What does a compiler do?
7. What part does the object program play in the compilation and execution processes?
8. Name the four basic ways of structuring statements in C++ and other languages.
9. What are the six basic components of a computer?
10. What is the difference between hardware and software?
11. In what regard is theft of computer time like stealing a car? How are the two crimes different?
12. What is the divide-and-conquer approach?

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\* Paycheck program

This program computes an employee's wages for the week \*/

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include <iostream> // standard input-output stream

using namespace std; // provides a scope to the identifiers

void CalcPay( float, float, float& );

const float max\_hours = 40.0; // normal hours worked

const float overtime = 1.5; // overtime pay rate factor

int main()

{

 float payRate, hours, wages; /\* Employee's pay rate, hours worked, and wages earned \*/

 int empNum; // Employee ID number

 cout << "Enter employee number: "; // Prompt

 cin >> empNum; // Read employee ID Number

 cout << "Enter pay rate: "; // Prompt

 cin >> payRate; // Read hourly pay rate

 cout << "Enter hours worked: "; // Prompt

 cin >> hours; // Read hours worked

 CalcPay(payRate, hours, wages); // Compute wages

 cout << "----- Output Result -----" << endl << "Employee No.: " << empNum << endl

 << "Pay rate: $" << payRate << endl

 << "Hours: " << hours << endl

 << "Wages: $" << wages << endl; // Output result to screen

 return 0; // Indicate successful completion

}

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void CalcPay( float payRate, float hours, float& wages )

/\* CalcPay computes wages from the employee's pay rate and the hours worked, taking overtime into account \*/

{

 if (hours > max\_hours) // overtime?

 wages = (max\_hours \* payRate) + (hours - max\_hours) \* payRate \* overtime; // Yes

 else

 wages = hours \* payRate; // No

}

C++

* + - * 1. What are similarities and differences do you see with Python and C++?
				2. How are comments shown?
				3. How is a block of a program shown?
				4. How are variables written?
				5. How are input/output written?
				6. How do you show successful completion of a program?