APCSP Name _____

Semester 1 Final Exam Information

The APCSP Semester 1 Final Exam will consist of two (2) parts:

1. Basic App Creation
2. Traditional" Paper Exam

This final exam is intended to mirror the format of the AP exam administered at the end of the year which also has two parts--Performance Tasks and End-of-Course Exam.

The details of each part are outlined below:

Semester 1 Final Exam (20% of Semester 1 Grade)		
Part 1—App Creation		**Part 2—"Traditional" Paper Exam**
• 40%	Weight/Percentage of Final Exam	• 60%
• See attached outline and activity guide • Written reflection completed during final exams *(9th per Jan 16, 2nd per. Jan 17, 3rd per. Jan 18)*	Requirements	• ~40 Multiple Choice Questions covering Units 1, 2, 3, 5—*see topics below** • Bring a pen/pencil • No outside notes/resources allowed
• App must be submitted **by 8AM Jan 17**	Due Date	• Taken in-class during final exams *(2nd per. Jan 17, 3rd per. Jan 18, 9th per. Jan 16)*

Semester 1 Topics Covered on Exam*

Unit 1—The Internet
- Chapter 1: Representing and Transmitting Information
- Chapter 2: Inventing the Internet

Unit 2—Digital Information
- Chapter 1: Encoding and Compressing Complex Information

Unit 3—Algorithms and Programming
- Chapter 1: Programming Languages and Algorithms

Unit 5—Buildings Apps
- Chapter 1: Event Driven Programming

UNIT 1 VOCABULARY

Lesson 1: Personal Innovations

- **Innovation**: A novel or improved idea, device, product, etc, or the development thereof.

Lesson 2: Sending Binary Messages

- **Binary**: A way of representing information using only two options.
- **Bit**: A contraction of "Binary Digit". A bit is the single unit of information in a computer, typically represented as a 0 or 1.

Lesson 3: Sending Binary Messages with the Internet Simulator

- **Bandwidth**: Transmission capacity measure by bit rate
- **Bit rate**: (sometimes written bitrate) the number of bits that are conveyed or processed per unit of time. e.g. 8 bits/sec.
- **Latency**: Time it takes for a bit to travel from its sender to its receiver.
- **Protocol**: A set of rules governing the exchange or transmission of data between devices.

Lesson 7: Encoding and Sending Formatted Text

- **ASCII**: ASCII - American Standard Code for Information Interchange. ASCII is the universally recognized raw text format that any computer can understand.
- **code**: (v) to write code, or to write instructions for a computer.

Lesson 8: The Internet Is for Everyone

- **IETF**: Internet Engineering Task Force - develops and promotes voluntary Internet standards and protocols, in particular the standards that comprise the Internet protocol suite (TCP/IP).
- **Internet**: A group of computers and servers that are connected to each other.
- **Net Neutrality**: the principle that all Internet traffic should be treated equally by Internet Service Providers.

Lesson 9: The Need for Addressing

- **IP Address**: A number assigned to any item that is connected to the Internet.
- **Packets**: Small chunks of information that have been carefully formed from larger chunks of information.
- **Protocol**: A set of rules governing the exchange or transmission of data between devices.

Lesson 10: Routers and Redundancy

- **Network Redundancy**: having multiple backups to ensure reliability during cases of high usage or failure
- **Router**: A type of computer that forwards data across a network

Lesson 11: Packets and Making a Reliable Internet

- **TCP**: Transmission Control Protocol - provides reliable, ordered, and error-checked delivery of a stream of packets on the internet. TCP is tightly linked with IP and usually seen as TCP/IP in writing.

Lesson 12: The Need for DNS

- **DNS**: The service that translates URLs to IP addresses.

Lesson 13: HTTP and Abstraction on the Internet

- **HTTP**: HyperText Transfer Protocol - the protocol used for transmitting web pages over the Internet
- **URL**: An easy-to-remember address for calling a web page (like www.code.org).

UNIT 2 VOCABULARY

Lesson 2: Text Compression

- **Heuristic**: a problem solving approach (algorithm) to find a satisfactory solution where finding an optimal or exact solution is impractical or impossible.
- **Lossless Compression**: a data compression algorithm that allows the original data to be perfectly reconstructed from the compressed data.

Lesson 3: Encoding B&W Images

- **Image**: A type of data used for graphics or pictures.
- **metadata**: is data that describes other data. For example, a digital image my include metadata that describe the size of the image, number of colors, or resolution.
- **pixel**: short for "picture element" it is the fundamental unit of a digital image, typically a tiny square or dot which contains a single point of color of a larger image.

Lesson 4: Encoding Color Images

- **Hexadecimal:** A base-16 number system that uses sixteen distinct symbols 0-9 and A-F to represent numbers from 0 to 15.
- **RGB**: the RGB color model uses varying intensities of (R)ed, (G)reen, and (B)lue light are added together in to reproduce a broad array of colors.

Lesson 5: Lossy Compression and File Formats

- **Lossy Compression**: (or irreversible compression) a data compression method that uses inexact approximations, discarding some data to represent the content. Most commonly seen in image formats like .jpg.

Lesson 6: Practice PT - Encode an Experience

- **Abstraction**: Pulling out specific differences to make one solution work for multiple problems.

UNIT 3 VOCABULARY

Lesson 2: The Need for Algorithms

- **Algorithm**: A precise sequence of instructions for processes that can be executed by a computer.
- **High Level Programming Language**: A programming language with many commands and features designed to make common tasks easier to program. Any high level functionality is encapsulated as combinations of low level commands.
- **Low Level Programming Language**: A programming language that captures only the most primitive operations available to a machine. Anything that a computer can do can be represented with combinations of low level commands.

Lesson 3: Creativity in Algorithms

- **Algorithm**: A precise sequence of instructions for processes that can be executed by a computer
- **Iterate**: To repeat in order to achieve, or get closer to, a desired goal.
- **Selection**: A generic term for a type of programming statement (usually an if-statement) that uses a Boolean condition to determine, or select, whether or not to run a certain block of statements.
- **Sequencing**: Putting commands in correct order so computers can read the commands.

Lesson 4: Using Simple Commands

- **Turtle Programming**: a classic method for learning programming with commands to control movement and drawing of an on-screen robot called a "turtle". The turtle hearkens back to early implementations in which children programmed a physical robot whose dome-like shape was reminiscent of a turtle.

Lesson 5: Creating Functions

- **Abstraction**: Pulling out specific differences to make one solution work for multiple problems.
- **Function**: A piece of code that you can easily call over and over again.

Lesson 6: Functions and Top-Down Design

- **Top Down Design:** a problem solving approach (also known as stepwise design) in which you break down a system to gain insight into the sub-systems that make it up.

Lesson 7: APIs and Using Functions with Parameters

- **API**: a collection of commands made available to a programmer
- **Documentation**: a description of the behavior of a command, function, library, API, etc.
- **Library**: a collection of commands / functions, typically with a shared purpose
- **Parameter**: An extra piece of information that you pass to the function to customize it for a specific need.

Lesson 9: Looping and Random Numbers

- **For Loop**: A particular kind of looping construct provided in many languages. Typically, a for loop defines a counting variable that is checked and incremented on each iteration in order to loop a specific number of times.
- **Loop**: The action of doing something over and over again.

UNIT 5 VOCABULARY

Lesson 1: Introduction to Event-Driven Programming

- **Callback function**: a function specified as part of an event listener; it is written by the programmer but called by the system as the result of an event trigger.
- **Event**: An action that causes something to happen.
- **Event-driven program**: a program designed to run blocks of code or functions in response to specified events (e.g. a mouse click)
- **Event handling**: an overarching term for the coding tasks involved in making a program respond to events by triggering functions.
- **Event listener** : a command that can be set up to trigger a function when a particular type of event occurs on a particular UI element.
- **UI Elements**: on-screen objects, like buttons, images, text boxes, pull down menus, screens and so on.
- **User Interface**: The visual elements of an program through which a user controls or communications the application. Often abbreviated UI.

Lesson 2: Multi-Screen Apps

- **Debugging**: Finding and fixing problems in an algorithm or program.

Lesson 4: Controlling Memory with Variables

- **Data Type**: All values in a programming language have a "type" - such as a Number, Boolean, or String - that dictates how the computer will interpret it. For example 7+5 is interpreted differently from "7"+"5"
- **Expression**: Any valid unit of code that resolves to a value.
- **Variable**: A placeholder for a piece of information that can change.

Lesson 5: Building an App: Clicker Game

- **==** The equality operator (sometimes read: "equal equal") is used to compare two values, and returns a Boolean (true/false). Avoid confusion with the assignment operator "=",
- **Global Variable**: A variable whose scope is "global" to the program, it can be used and updated by any part of the code. Its global scope is typically derived from the variable being declared (created) outside of any function, object, or method.
- **If-Statement**: The common programming structure that implements "conditional statements".
- **Local Variable**: A variable with local scope is one that can only be seen, used and updated by code within the same scope. Typically this means the variable was declared (created) inside a function -- includes function parameter variables.
- **Variable Scope**: dictates what portions of the code can "see" or use a variable, typically derived from where the variable was first created. (See Global v. Local)

Lesson 6: User Input and Strings

- **Concatenate**: to link together or join. Typically used when joining together text Strings in programming (e.g. "Hello, "+name)
- **String**: Any sequence of characters between quotation marks (ex: "hello", "42", "this is a string!").

Lesson 7: If-statements unplugged

- **Conditionals**: Statements that only run under certain conditions.
- **Selection**: A generic term for a type of programming statement (usually an if-statement) that uses a Boolean condition to determine, or select, whether or not to run a certain block of statements.

Lesson 8: Boolean Expressions and "if" Statements

- **Boolean**: A single value of either TRUE or FALSE
- **Boolean Expression**: in programming, an expression that evaluates to True or False.
- **Conditionals**: Statements that only run under certain conditions.
- **If-Statement**: The common programming structure that implements "conditional statements".

UNIT 3 PROGRAMMING COMMANDS

- moveForward
- penUp
- turnLeft
- penDown
- function myFunction() { // function body, including optional "return" command. }
- arcLeft
- arcRight
- penWidth
- penRGB
- turnTo
- moveTo
- penColor
- randomNumber min/max
- function myFunction(n){ //code }

UNIT 5 PROGRAMMING COMMANDS

- setSize
- onEvent(id, type, function(event)){ ... }
- setPosition
- setScreen
- console.log
- write
- randomNumber
- num1 * num2;
- value1 + value2;
- num1 / num2;
- num1 - num2;
- var x = promptNum("Enter a value");
- x = __;
- var x = "__";
- var x = __;
- setText(id, text)
- __ == __
- getText(id)
- str.toLowerCase
- str.toUpperCase
- if( ) { //code }
- if ( ) { // if code } else { // else code }
- __ == __
- __ != __
- __ < __
- __ <= __
- __ > __
- __ >= __
- !__
- __ && __
- __ || __
- rgb(r, g, b, a)