Computational Thinking Competition Project Ideas

GRADES 10-12

Computational Thinking is a problem-solving process that includes the following characteristics:

- Formulating problems in a way that enables us to use a computer and other tools to help solve them (i.e. A real world issue that needs to be addressed).
- Logically organizing and analyzing data i.e. representing data through abstractions such as models and simulations (i.e. Modeling)
- Identifying, analyzing, and implementing possible solutions with the goal of achieving the most efficient and effective combination of steps and resources. Automating solutions through a series of ordered steps (i.e. Coding)
- Testing and Generalizing and transferring this problem solving process to a wide variety of problems.

There are no limits on the problem areas and topics could be any of (and not limited to):

- Example Problem Areas
  - Computational Biology
  - Cryptography & Encryption algorithmic design
  - Big Data algorithms
  - Physics, Chemistry, Math, Biology, ...
  - Any engineering area
  - Social media

- Example Implementation platforms
  - Mobile Apps
  - Web Apps
  - Desktop Apps
  - Games
  - Any language (python/java/c# etc)

Examples

The following examples are from past contests.

1. Encrypted Storage [2017 project]
2. Computer Adder Simulator [2017 project]
3. Quiz Analysis [2017 project]

1) Encrypted Storage

This was a password protected encryption storage directory accessed using the command-line. This was implemented using Java.

- Directory operations
  - mkdir [dirname] creates a new directory
  - dldr [dirname] deletes directory
  - dpdr [dirname] duplicates dir and all of its contents
  - cldr [dirname] deletes all files in dir

- File operations
  - reda [filename] decrypts and reads the file
  - redx [filename] reads undecrypted file
  - delt [filename] deletes the file
  - wrta [filename] creates or writes encrypted content to file
  - wrtx [filename] creates or writes unencrypted content to file
  - clre [filename] clears all text in file

Encrypted Storage cont'd

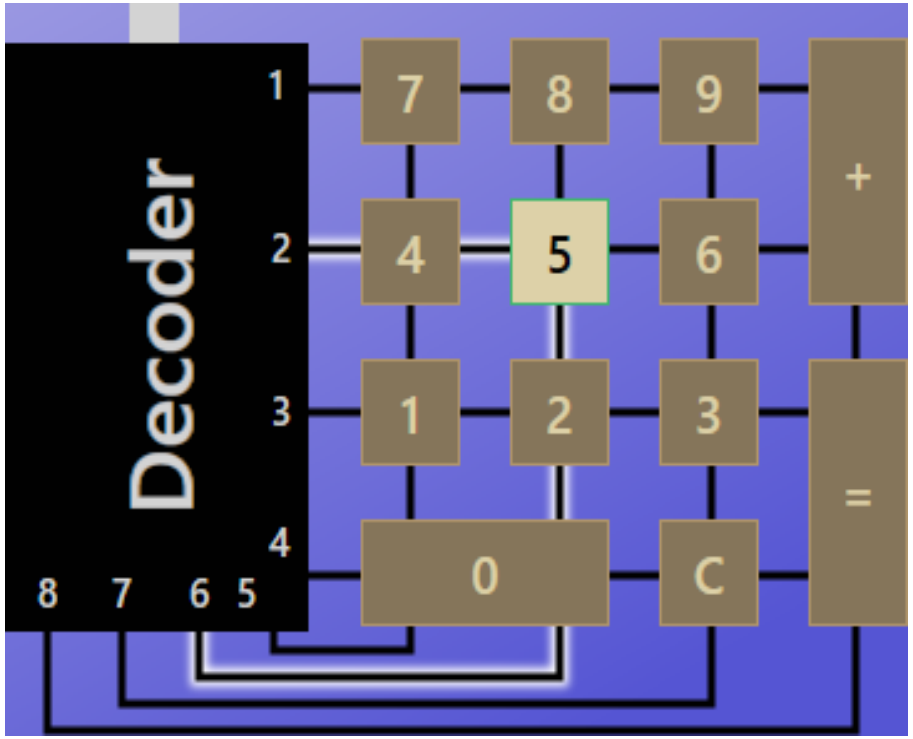
- Misc commands
  - dupl [file/dir] duplicate file or directory
  - renm [file/dir] rename file or directory
  - encr [text] encrypts text
  - decr [text] decrypts text
  - newP creates new Password
  - cred credits for people involved in development
  - info information or description of the tool
  - help list of commands available
  - exit exits the system

2) Computer Adder Simulator

- The goal was to demonstrate how pressing on a button translates to a number with matrix switch logic, more commonly called Crossbar Switch logic, and a visual display of 8-bit binary addition using logic gates with overflow detection.
- C# was used for the project's programming language. Microsoft Visual Studio 2015 Community was used as the development and compiler environment. C# combined with Windows Presentation Foundation (WPF) provides power and flexibility required for this project.

Crossbar Switch

The Crossbar Switch is a matrix of keys and wires utilizing open/closed switches allowing power through once pressed. Each press is recognized by a location (powered wires) on the matrix.

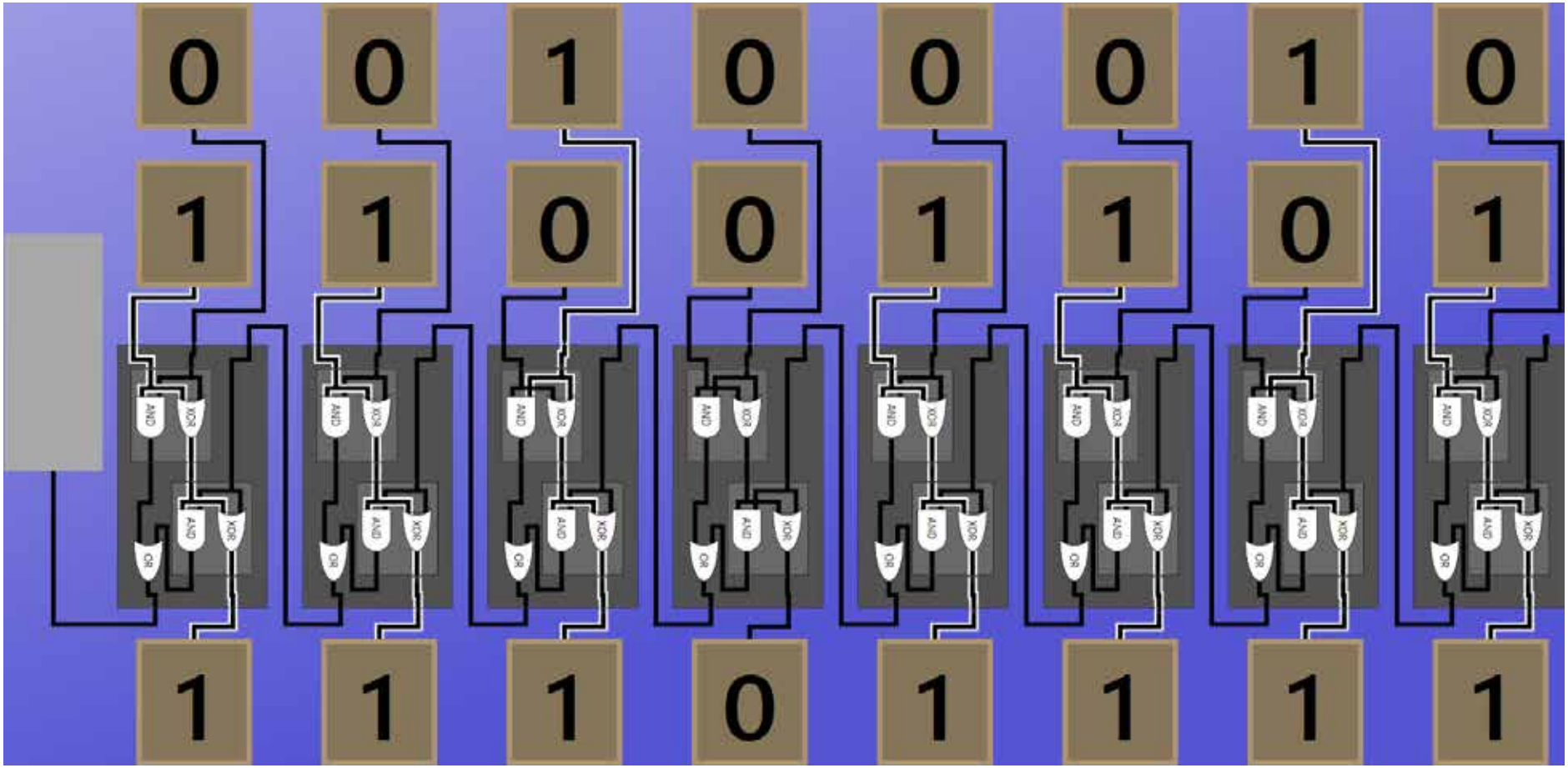


when '5' is pressed, power flows between pins 2 and 6, which the decoder then recognizes as '5' and translates it to a bytecode to send to the display.

8-bit Adder

The 8-Bit Adder is built using 8 Full Adders which take in 16 inputs or two numbers. The 8-Bit Adder used in this simulation is called a *ripple carry*.

8 bit adder at work



Number1 is 00100010 which is equal to 34

Number2 is 11001101 which is equal to 205.

Final sum is 11101111 which is 239.

3) Quiz Analysis

This was written in Java and the purpose was to allow creation of quiz questions and to generate statistical information from results of quizzes.

List of commands were:

- select (term1) [term2] [term3]
- clear
- ask (percentCorrect)
- import (path)
- help

Code Organization

Hierarchy For Package version_five

Class Hierarchy

- java.lang.Object
  - version_five.**AbstractFileManager**
    - version_five.**DefaultFileManager**
  - version_five.**CharTree**<E>
  - version_five.**DefaultAnalyzer** (implements version_five.IAnalyzer)
  - version_five.**DefaultEngine** (implements version_five.IEngine)
  - version_five.**DefaultQuestion** (implements version_five.IQuestion)
  - version_five.**Stat**
  - version_five.**TestDefaultAnalyzer**
  - version_five.**TestDefaultQuestion**
  - version_five.**TestType**
  - version_five.**TestType.Test1234** (implements version_five.IQuestion)
  - version_five.**Type**

Interface Hierarchy

- java.lang.Comparable<T>
  - version_five.**IQuestion**
- version_five.**IAnalyzer**
- version_five.**IEngine**