CptS 122 – Data Structures
Programming Assignment 4: Implement Yu Gi Oh! Card Game

Assigned: Monday, June 1st, 2020
Due: Monday, June 8th, 2020 @ midnight

I. Learner Objectives:

At the conclusion of this programming assignment, participants should be able to:

- Design, implement and test classes in C++
- Apply class templates in C++
- Declare and define *constructors*
- Declare and define *destructors*
- Compare and contrast *public* and *private* access specifiers in C++
- Describe what is an *attribute* or data member of a class
- Describe what is a *method* of a class
- Apply and implement *overloaded* functions
- Distinguish between pass-by-*value* and pass-by-*reference*
- Discuss *classes* versus *objects*
- Implement a *queue* class
- Implement a *stack* class
- Utilize an array to manage data
- Read and write files in C++

II. Prerequisites:

Before starting this programming assignment, participants should be able to:

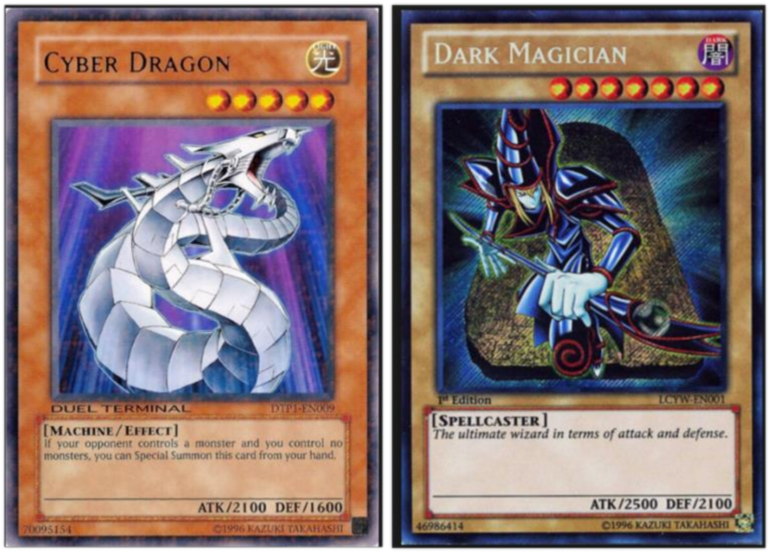
- Analyze a basic set of requirements for a problem
- Compose basic C++ language programs
- Create basic test cases for a program
- Apply arrays, strings, and pointers

III. Overview & Requirements:

Yu-Gi-Oh! is a trading card game where the objective is to defeat your opponent by reducing their Life Points to zero.

Understanding the Cards:

Monster cards are summoned to attack your opponent's Life Points and defend your own. They are usually orange (effect) or yellow (normal) in color, but there are many other colors as well. Monsters have levels, ranging from 1-12, which are indicated by the stars along the top, and a symbol in the top right corner indicating Attribute. The Attack and Defense stats are listed as **ATK** and **DEF** along the bottom.



What you will implement:

Monster Cards: (*Node Struct*)

a. **Name**
b. **Attack Points** – # of points to take from life points
c. **Defense Points** – # of points to keep subsequent attack
d. **Type** – monster type gives extra points to the ATK or DEF points

Implementation:

1. Each player can start out with a set of cards (*the deck*):
    a. Player adds card individually
    b. Player gets a set of predefined cards
2. Players can go into battle
    a. Player starts out with **8000** life points
    b. Coin Toss to see who goes first player 1 or player 2
    c. Each Player chooses **5** cards from the *deck* into their respective *hands*
        i. Player can only choose from the **top** of the deck
    d. Each player will place one card of their choosing in **defense mode (queue)**

  e. Each player will place one card of their choosing in **attack mode (queue)**
  f. Each player will choose to activate a card from one of the queues
     i. If one player activates the attack card and the other activates a defense card. Attack wins and defense gets the point taken away
        1. New Life points = Life points - (ATK – DEF)
     ii. If both players choose defense
        1. Cards get discarded – not life points are taken
     iii. If both players choose attack
        1. One with greater ATK wins round
        2. New Life points = Life points - (ATK – ATK)
  g. After the first round
     i. 5 cards should be in the players hand (array) at all times retrieved from the deck (stack)
     ii. Two more cards are placed in attack and defense mode (queues)
     iii. Cards must be activated in queue mode

3. Battle ends when a player has no more Life Points

Example of output:

```
Choose an option from the following:

    1. Create Players
    2. Add Cards to Player
    3. Battle Other Player

    Input: 3

    Round 1:

    Player 1 chooses Cyber Dragon in ATK
    mode

Player 2 chooses Dark Magician in DEF
mode

Player 2 losses 0 life points

Player 1: 8000 LP

Player 2: 8000 LP

Round 2: ....
```

- Options 1 creation of a player should populate both players with a 30 card deck
- Option 2 if a certain player wants to add more specific cards to their 30 card deck

Note: option 1 in the menu needs to be run by the players before trading or battling commences

- You will have two classes: Stack class and Queue class.
- The *main.cpp* will implement the menu above.

IV. Grading Guideline:

This assignment is worth 100 points. Your assignment will be evaluated based on a successful compilation and adherence to the program requirements. We will grade according to the following criteria:

- Menu – 10 points
- Coding functionality (organization of code) – 10 points
- Gaming aspect – 70 points
    - Adding cards to a deck – 10 points
    - Adding to the hand – 10 points
    - Adding to defense queue – 10 points
    - Adding to attack queue – 10 points
    - Life Points Calculation – 10 points
    - Winner Decision – 20 points
- Quit Game – 10 points

V. Submitting Assignments:

1. Must submit your assignment in a zip file through ***blackboard***.
2. Your project must contain at least two header files (.h files) and three C++ source files (which must be .cpp files).
3. Your project must build properly. The most points an assignment can receive if it does not build properly is 65 out of 100.

Take the cards from the hello.txt
- Put each card and num into the stack
- Have player1 and player2 get 3 cards
- Each player puts their cards into the queue

```
//File Input and Output
    cout << "\n\nFile Time";
    ifstream iFile("hello.txt");
    ofstream oFile("result.txt", ios::app);

    string name;
    string number;
    int count = 1;
    while(!iFile.eof())
    {
        getline(iFile, name, ',');
        getline(iFile, number, ',');
        oFile << input << endl;
        cout << "\nConfirmation\t" << input;
    }
    cout << "\nCount is " << count;
```