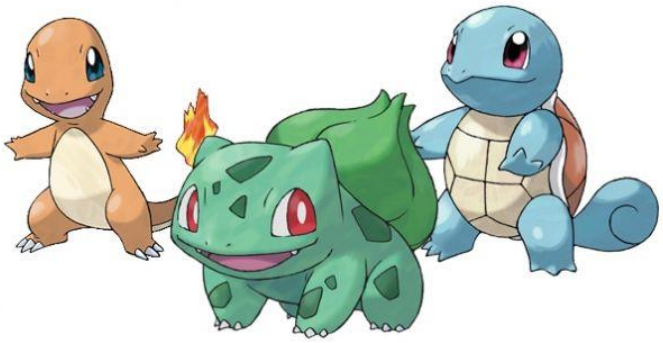


PROJECT 2

<Pokémon Game>

CONNOR KELLEY

06/08/15

CSC 5 - 43952

Introduction

Title: Pokémon Game

Pokémon is a turn-based battle game that involves attacking or running if you need to. In this game, the object is to battle different monsters to gain experience and eventually level up if enough experience is accumulated. The ultimate goal of the game is to get to a high enough level, and to get your Pokémon to a strong enough strength to be able to take on the world of Pokémon with ease. This is a turn based battle game. The user is allowed to opportunity to attack the foe but then the foe is allowed to attack the user back directly afterward. The first player to have their health fall below the fatal level of "0" loses. The object of the game then, is to get your opponent's health to fall below zero before they get yours to fall below zero.

Summary

Project Size: 237 Lines

Number of Variables: 15

Number of Functions: 6 + main

In my game program I was able to use multiple data types including strings, integers, and floats. I utilized the file save and load input/output functions programmed into the fstream system library. I also tapped into the ctime and cstdlib to create random numbers to calculate attack damage. On multiple occasions I used if statements to determine and calculate how powerful of an enemy the user would face. Additionally, I used loops to keep the user engaged in the menu selection until they opted to exit the game entirely. I used six different functions to break up some of the menu and introductory dialogue in the game. Doing this, I experienced some difficulty passing data through functions and having the data return back to main. In my program, there are a lot of improvements that I will need to make in order to improve the way the gameplay functions. Some algorithms that I used to determine enemy health and levels are sometimes wonky, often giving out too high of numbers. I will modify this in the second version as well as add more functionality and dynamic elements to the battle gameplay itself.

This project took me well over a week to complete. I have zero experience programming and developing games so this was definitely a wakeup call. Many challenges that I faced I used the Gaddis book for reference or I searched online, usually stackoverflow.com or cplusplus.com.

Description

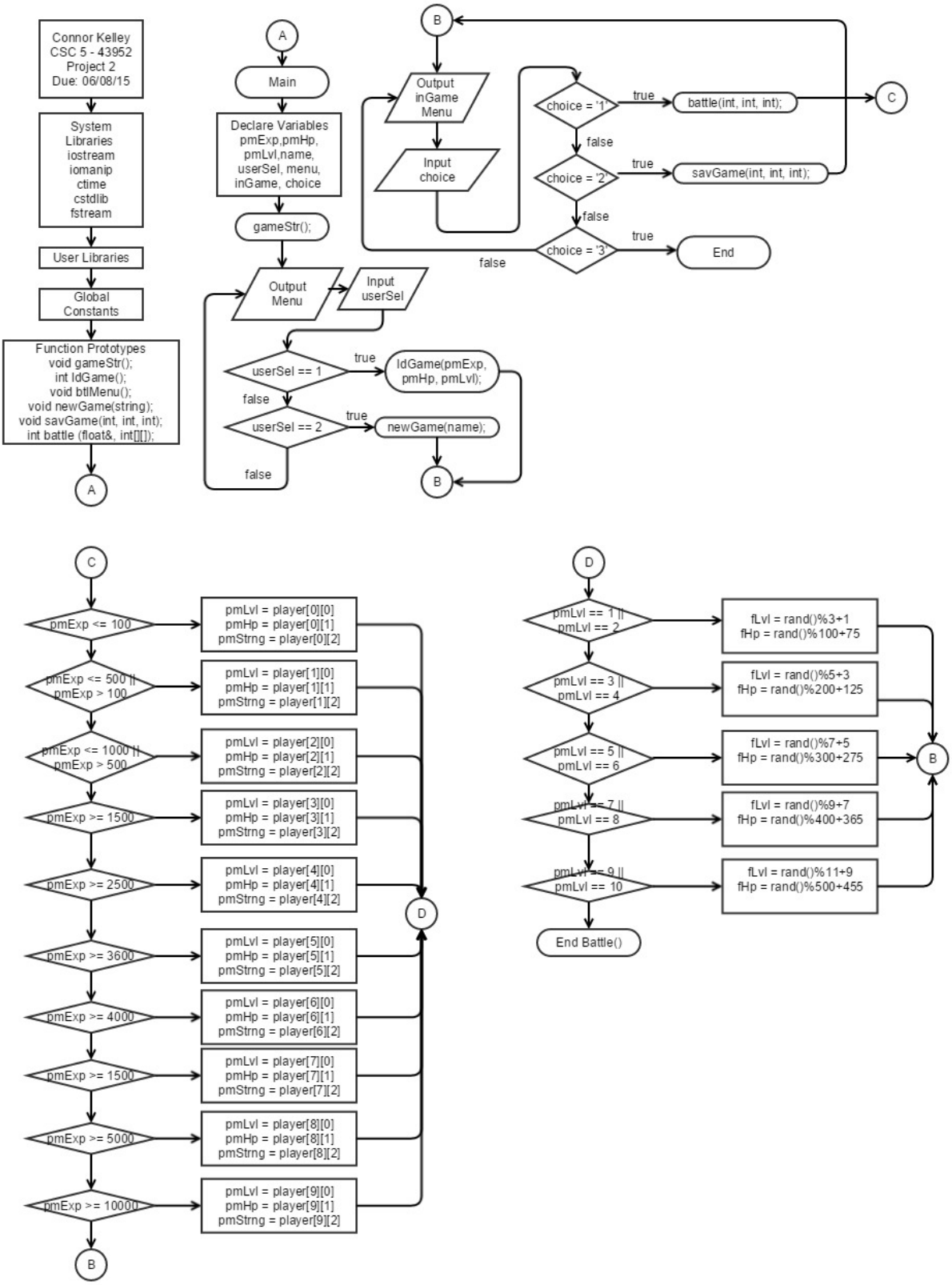
The purpose of my program is to create a function game which engages the user in combat with a random number generated PC enemy.

Objectives

/* Project Requirements:
 * Minimum of 250 lines of code X
 * Functions X
 * One Dimensional Array X
 * Two Dimensional Array X
 * Pass arrays between functions X
 * Function Pass by Value X
 * Function Pass by reference X
 * Defaulted Parameters X
 * Returning primitive data types X
 * Output format X
 * Read and write to file X
 * Sorting game players and searching O
 */

Flow Chart

Pokemon Project 2



Pseudo Code

```
//System Libraries
//c standard
//input output stream
//input output manipulation
//c time
//file stream
//User Libraries
//Global Constants
//Function Prototypes
   //Return Experience Points attained
   //Return information about pokemon

//Execution Begins Here!
   //Declare Player Variables
   //Pokemon Experience
        // Pokemon Hit points
            //Pokemon Level
            //Player Name
   //Other Variables
   //Start Game
   //Game Menu to load or start new game
//If selection is 1 then load a previous game
//If selection is 2 then start a new game
//If selection does not match case, notify and print the menu again
   //Start the in-game menu...
   //If user select 1 then start the battle sequence function
            //if the user selects 2 then save the game
            //if the user selects 3 then exit the program
   //Exit Stage Right!

//Function for Game start
   //Display starting content (TITLE SCREEN)
   //Wait for the user to press 'Enter'
//Function to Load Previous Game
   //Must load the name of the player
            //save all variables related to pokemon
//Function to Create New Game
   //Introduction to game and game play
   //Player enters their name
   //Opening Dialogue... Information about game play
   //Collect the name of the player
   //Eventually insert information about the starter Pokemon?
   //Would this be useful? Maybe some stats?

//Function to Save Previous Game
//Saves:
   //Name
   //Pokemon level

//Function to initiate battle sequence...
```

```
   //Initialize random number seed
   //Declare Variables
//Foe's HP
//Foe's Strength Value
//Foe's Level
//Pokemon Strength
  //Calculate the enemy's level,hit points
   //Output Battle Sequence...
   //Output Enemy Stats!
//Output battle selection
//if user enters 1 then attack!
            //Attack the enemy!
//if user enters 2 then roll to run away safely
   //Determine win or lose
//if lose then decrease level
//if win then increase experience
   //Exit Stage Right!
```

Major Variables

Type	Variable Name	Description	Location
Integer	pmExp	Pokémon Experience	Int main() Int battle()
	pmHp	Pokémon Hit Points – The health of the Pokémon. The larger the value the longer it will take to defeat it.	Int main() Int battle()
	pmLvl	Pokémon Level – this value is then used to manipulate attack damage and health	Int main() Int battle()
	fHp	The enemy player's hit points	int battle()
	fStrng	The enemy player's strength value, or attack value.	int battle()
	fLvl	The enemy player's level. Used in an algorithm to construct the hit point and strength or attack value of their Pokémon.	int battle()
	pmStrng	The user Pokémon's strength value, or attack value.	int battle()
	userSel	User Selection – used to guide menu algorithms	Int main()
	randNum	A randomly generated number according to ctime library.	int battle()
const int	SIZE	The size of the array to store random health values for the enemy players fHp.	int battle()
	ROWS	A constant integer that dictates the amount of rows in the player data table	int battle() int main()
	COLS	A constant integer that dictates the amount of columns in the player data table	int battle() int main()

int[]	hpAry[]	This array stores the randomly generated enemy hp values.	int battle()
int[][]	player[][]	This array stores the hit points and the level of the player—this is calculated according to experience gathered. This array is passed from main to battle to main etc.	int main() int battle()
Bool	menu	Menu driven algorithm	Int main()
	inGame	Menu driven algorithm	Int main()
String	name	Used to store the name of the user	Int main() void savGame(string)
Char	choice	User choice used to operate switch	int main()
	select	User select used to operate switch	int battle()
ifstream	inFile	Used to load the Pokémon's experience, hit points, and level from a saved file.	int ldGame() int main()
ofstream	myGame	Used to save the Pokémon's experience, hit points, and level to a file.	int savGame()

Player Array Details

Level	Hit Points	Attack
1	100	rand()%10+1
2	200	rand()%15+5
3	300	rand()%20+10
4	400	rand()%25+15
5	500	rand()%30+20
6	600	rand()%35+25
7	700	rand()%40+30
8	800	rand()%45+35
9	900	rand()%50+40
10	1000	rand()%55+45

Program

```
/*
 * File:   main.cpp
 * Author: Connor Kelley
 * Created on May 19, 2015, 11:13 AM
 * Purpose: Improve the functionality and expand upon my original game design
 *          for the game of Pokemon -- a turn based strategy battle game
 */

//System Libraries
#include <cstdlib>  //c standard
#include <iostream> //input output stream
#include <iomanip>  //input output manipulation
#include <ctime>    //c time
#include <fstream>  //file stream
using namespace std;

//User Libraries

//Global Constants

//Function Prototypes
void gameStr();
void ldGame(string&, float&);
void btlMenu();
void newGame(string&);
void savGame(string, float);
int battle(float&, int[10][3]);
   //Return Experience Points attained
   //Return information about pokemon

//Execution Begins Here!
int main(int argc, char** argv) {
   //Declare Player Variables
   float   pmExp   = 0;  //Pokemon Experience
   string  name;   //Player Name

   //Menu Variables
   int userSel;
   bool menu   = true,
        inGame = true;
   char choice;

   //Start Game
   gameStr();

   //Game Menu to load or start new game
   do{
      cout<<"********     MENU     *********"<<endl;
      cout<<"Enter 1 to load a previous game"<<endl;
      cout<<"Enter 2 to  start  a  new  game"<<endl;
```

```
   cout<<"*******************************"<<endl;
   cin>>userSel;
   if (userSel == 1){
      //Load an already saved game from a file into the game
      ldGame(name, pmExp);
      menu = false;
   } else if (userSel == 2){
      //Initiate the new game sequence to gather player name and introduce game
      newGame(name);
      menu = false;
   } else {
      //Catch user errors
      cout<<"You did not enter a valid menu selection"<<endl;
   }
} while(menu);

//Calculate Level -- based on experience points gathered
//Level then determines the hit points and attack power respectively

//Pass this to the battle function instead of declaring it in main then passing single variables
//-- pass the whole thing
const int ROWS = 10;    //Row total of the array
const int COLS = 3;     //Column total of the array
                  //  --Includes the Level, Hit Points(HP), and Attack Power
int player[ROWS][COLS] =   {{1, 100, rand()%10+1},
                     {2, 200, rand()%15+5},
                     {3, 300, rand()%20+10},
                     {4, 400, rand()%25+15},
                     {5, 500, rand()%30+20},
                     {6, 600, rand()%35+25},
                     {7, 700, rand()%40+30},
                     {8, 800, rand()%45+35},
                     {9, 900, rand()%50+40},
                     {10, 1000, rand()%55+45}};

//Start the in-game menu...
while(inGame){
   cout<<endl<<name<<", what would you like to do?"<<endl;
   cout<<"[1.] Battle a Pokemon"<<endl;
   cout<<"[2.] Save the game"<<endl;
   cout<<"[3.] Exit the game"<<endl;
   cin>>choice;
   switch (choice){
      case '1':{
         battle(pmExp, player);
         break;
      }
      case '2':{
         savGame(name, pmExp);
         cout<<"Your game has saved!"<<endl;
         break;
      }
      case '3':{
```

```
            exit(0);
            inGame = false;
            break;
          }
        }
    }
    //Exit Stage Right!
    return 0;
}

//Function for Game start
void gameStr(){
    //Display starting content (TITLE SCREEN)
    cout<<"************************\n"
        <<"Pokemon by Connor Kelley\n"
        <<"  Press  [Enter]  to start \n"
        <<"************************\n";
    //Wait for the user to press 'Enter'
    cin.ignore();
}

//Function to Save Previous Game
//Saves:
    //Name
    //Experience
void savGame(string name, float pmExp){
//    int tempExp;
//    string tempNm;
//
//    ifstream inFile;
//    inFile.open("savedGame.txt");
//    getline(inFile, tempNm);
//    inFile>>pmExp;
//    inFile.close();

    ofstream myGame;
    myGame.open("savedGame.txt");
    myGame<<name;
    myGame<<endl<<pmExp;
    myGame.close();
}
//Function to Load Previous Game
void ldGame(string &name, float &pmExp){
    ifstream inFile;
    inFile.open("savedGame.txt");
    getline(inFile, name);
    inFile>>pmExp;
    inFile.close();
}

//Function to Create New Game
    //Introduction to game and game play
    //Player enters their name
```

```
    //Name is passed back into main
void newGame(string &name){
  //Opening Dialogue... Information about game play
  cout<<endl<<"****************************************\n"
        "Welcome to the wonderful game of pokemon!"<<endl;

  //Collect the name of the player
  cout<<"Start by entering your name: ";
  cin.ignore();
  getline(cin, name);
  cout<<endl<<"Awesome name, "<<name<<"!"<<endl;
  cout<<"Here is some information about pokemon.\n"
  //Eventually insert information about the starter Pokemon?
  //Would this be useful? Maybe some stats?
      "Over time your pokemon will grow stronger the more\n"
      "you successfully defeat other foes in battle. You will\n"
      "be able to [attack] your foe while in battle, and may\n"
      "acquire new attacks after you gather enough experience.\n\n"
      "Press [enter] to continue..."<<endl;
  cin.ignore();
  cout<<"You will face many foes on your journey. Strong foes with\n"
      "higher [hp] values will yield more [exp]erience for your pokemon.\n"
      "If your pokemon dies on the field of battle your pokemon will lose a level...\n\n"
      "Press [enter] to continue..."<<endl;
  cin.ignore();
  cout<<"Be careful not to face creatures too powerful to defeat.\n"
      "Remember, you can always [Run] at any time in battle. If you\n"
      "choose to [Run], you will not gain any experience for that fight.\n\n"
      "Press [enter] to continue..."<<endl;
  cin.ignore();
  cout<<endl<<name<<", you are now ready to become a Pokemon trainer\n"
        "and embark on your journey into the land of pokemon!"<<endl;
}

//Function to initiate battle sequence...
int battle(float &pmExp, int player[10][3]){
  //Initialize random number seed
  srand(static_cast<unsigned int>(time(0)));
  //Declare Variables -- Get them ready for battle!
  int   fHp,   //Foe's HP
      fStrng, //Foe's Strength Value
      fLvl,   //Foe's Level
      newExp; //New Experience points earned
      fStrng = rand()%10+1; //Calculate how strong the foes attacks are
  int   pmLvl  = 1,  //Pokemon Level
      pmHp   = 0,  //Pokemon Hit Points
      pmStrng = 0;  //Pokemon Strength

  //Determine level of the player based on the accumulated experience
  //Determine Hit Points of the player based on the accumulated experience
  if(pmExp <= 100){
    pmLvl  = player[0][0]; //Level = 1
    pmHp   = player[0][1]; //HP   = 100
```

```
    pmStrng = player[0][2]; //Strength = rand()%10+1
} else if ((pmExp <= 500)&&(pmExp > 100)){
    pmLvl   = player[1][0]; //Level = 2
    pmHp    = player[1][1]; //HP    = 200
    pmStrng = player[1][2]; //Strength = rand()%15+5
} else if ((pmExp <= 1000)&&(pmExp > 500)){
    pmLvl   = player[2][0]; //Level = 3
    pmHp    = player[2][1]; //HP    = 300
    pmStrng = player[2][2]; //Strength = rand()%20+10
} else if (pmExp >= 1500){
    pmLvl   = player[3][0]; //Level = 4
    pmHp    = player[3][1]; //HP    = 400
    pmStrng = player[3][2]; //Strength = rand()%25+15
} else if (pmExp >= 2500){
    pmLvl   = player[4][0]; //Level = 5
    pmHp    = player[4][1]; //HP    = 500
    pmStrng = player[4][2]; //Strength = rand()%30+20
} else if (pmExp >= 3600){
    pmLvl   = player[5][0]; //Level = 6
    pmHp    = player[5][1]; //HP    = 600
    pmStrng = player[5][2]; //Strength = rand()%35+25
}else if (pmExp >= 3600){
    pmLvl   = player[6][0]; //Level = 7
    pmHp    = player[6][1]; //HP    = 700
    pmStrng = player[6][2]; //Strength = rand()%40+30
}else if (pmExp >= 3600){
    pmLvl   = player[7][0]; //Level = 8
    pmHp    = player[7][1]; //HP    = 800
    pmStrng = player[7][2]; //Strength = rand()%45+35
}else if (pmExp >= 3600){
    pmLvl   = player[8][0]; //Level = 9
    pmHp    = player[8][1]; //HP    = 900
    pmStrng = player[8][2]; //Strength = rand()%50+40
}else if (pmExp >= 3600){
    pmLvl   = player[9][0]; //Level = 10
    pmHp    = player[9][1]; //HP    = 1000
    pmStrng = player[9][2]; //Strength = rand()%55+45
}

//Calculate the enemy's level and hit points
if((pmLvl == 1)||(pmLvl == 2)){
    fLvl = rand()%3+1;
    fHp  = rand()%100+75;
} else if ((pmLvl == 3)||(pmLvl == 4)){
    fLvl = rand()%3+5;
    fHp  = rand()%200+125;
} else if ((pmLvl == 5)||(pmLvl == 6)){
    fLvl = rand()%5+7;
    fHp  = rand()%300+275;
} else if ((pmLvl == 7)||(pmLvl == 8)){
    fLvl = rand()%7+9;
    fHp  = rand()%400+365;
} else if ((pmLvl == 9)||(pmLvl == 10)){
```

```
   fLvl = rand()%9+11;
   fHp  = rand()%500+455;
}

//Output Battle Sequence...
cout<<endl<<"You've encountered a foe!..."<<endl;
//Output Enemy Stats!
cout<<"Enemy level   : "<<fLvl<<endl;
cout<<"Enemy HP      : "<<fHp<<endl;
do{
   char select;
   cout<<"What would you like to do?"<<endl;
   cout<<"[1.] Attack"<<endl;
   cout<<"[2.] Run   "<<endl;
   cin>>select;
   switch (select){
      case '1':{
         //Attack the enemy!
         int dmg = pmStrng*pmLvl;
         fHp -= dmg;
         cout<<"You attack the enemy!"<<endl;
         if(dmg > 8){
            cout<<"Critical Hit!!"<<endl;
         }
         cout<<"You hit the enemy for "<<dmg<<" damage!"<<endl;
         cout<<"The enemy HP is now at: "<<fHp<<endl<<endl;
         if(fHp > 0){
            cout<<"The enemy attacks you!"<<endl<<"You get hit for "<<fStrng<<" damage."<<endl;
            pmHp -= fStrng;
            cout<<"Your HP is now at: "<<pmHp<<endl;
         }
         break;
      }
      case '2':{
         int randNum = rand()%10+1;
         if(randNum > 5){
            cout<<"You ran away safely..."<<endl;
            fHp = -99;
         } else {
            cout<<"Oh no! You failed to run away!!"<<endl;
            cout<<"The enemy attacks you! You get hit for "<<fStrng<<" damage."<<endl;
            pmHp -= fStrng;
            cout<<fixed<<setprecision(2)<<showpoint;
            cout<<"Your HP is now at: "<<pmHp<<endl;
         }
         break;
      }
   }
} while ((fHp > 0)&&(pmHp > 0)&&(fHp != -99));

//Determine win or lose or run
if((pmHp <= 0)&&(fHp >= 0)&&(fHp != -99)){
   cout<<"You lost the battle."<<endl;
```

```
        if (pmLvl > 1){
            pmExp -= ((rand()%5+1)*fLvl);
            cout<<"You lost "<<pmExp<<" experience points!"<<endl;
            cout<<"Your current level is now at "<<pmLvl<<endl;
        } else cout<<"Your level is too low to lose experience!"<<endl;
    } else if ((fHp <= 0)&&(pmHp >= 0)&&(fHp != -99)){
        cout<<"CONGRATS! YOU WON THE BATTLE!"<<endl;
        newExp = (rand()%100+75)*(pmLvl);
        pmExp += newExp;
        cout<<"You gained "<<newExp<<" experience for winning!"<<endl;
    } else if (fHp = -99){
        cout<<"Because you ran away from combat,"<<endl;
        cout<<"You neither gained nor lost experience."<<endl;
    }

    //Exit Stage Right!
    return pmExp, pmHp, pmLvl;
}
```