**Project 1**

Title

**Warriors’ Extravaganza**

Course

**CSC-17A**

Section

**48130**

Due Date

**October 21, 2014**

Author

**Kevin Vo**

**Introduction**

Title: Warriors’ Extravaganza

Warriors’ Extravaganza is a Role-playing game (RPG) where the player would decide on their name and from there, fight until he/she is power enough to defeat the maniacal boss. The player first starts off with very little health and damage, making it impossible to combat the boss right away. Therefore, it is a necessity that multiple trials of killing lesser creatures (minions). While fighting the minions, the player is given very little information besides the enemy’s health along with their own. The amount of damage being given to them has to be notice and memorized by the player and that includes the boss. In addition, nine potions are given to permanently boost health and damage during the minion trials, but there is a risk of overdosing and dying when too much is taken. Finally, once strong enough and the boss is defeated or as long as the player survives up until that level, he/she has to option to store their stats into a text file so that it can be viewed later. Overall, the reason being doing this project and why it is important is because it allows people to temporarily immerse themselves in a different world with a more interesting back story.

**Summary**

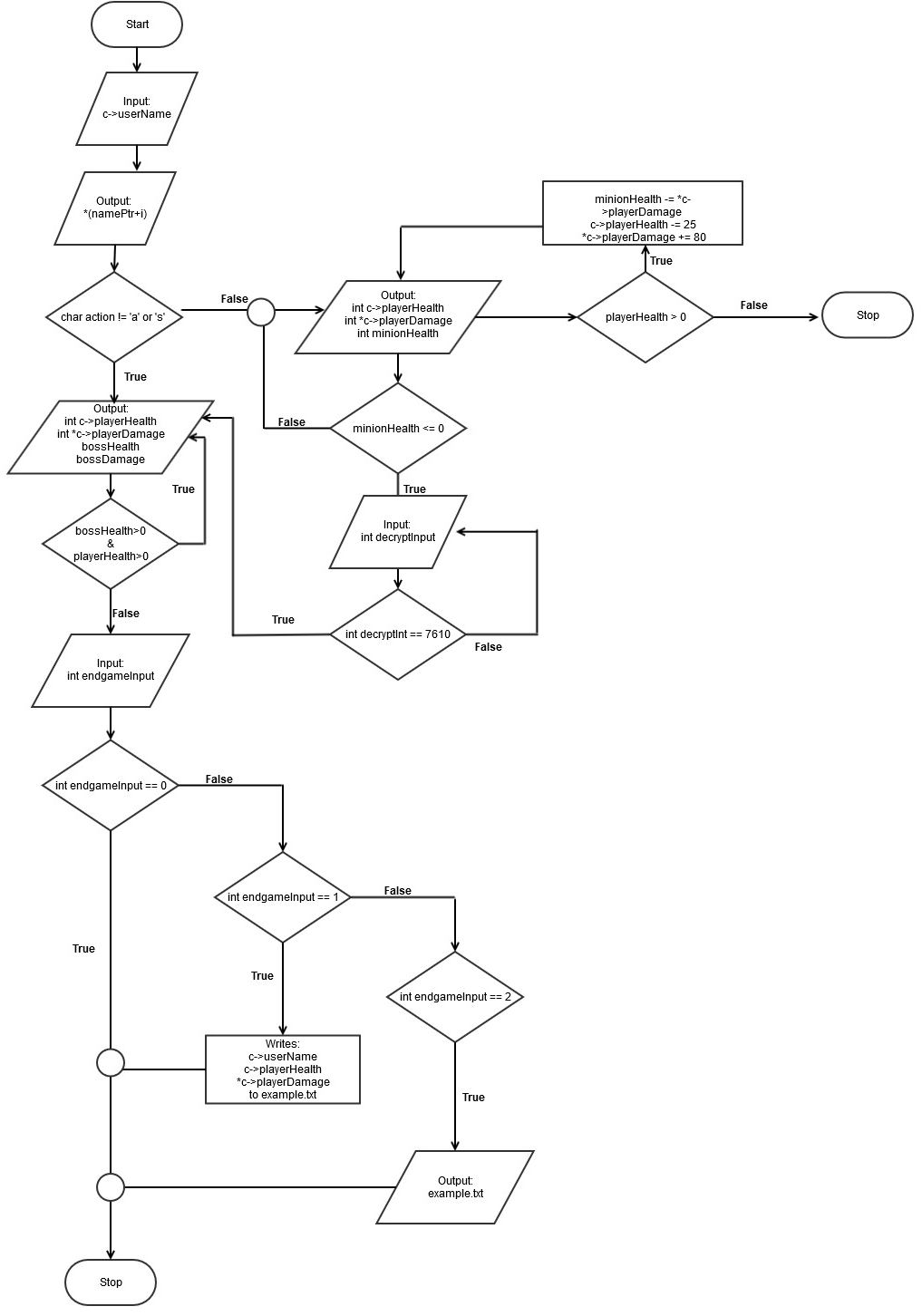
Project Size: 373 lines

The number of variables: ~20 variables

Structures: 1

At the beginning it was a fairly challenging project for me to picture out the layout for the game and to implement the new things that we have learned for this class. As a result, I wrote the program to the best of my knowledge from CSC-5 and then include that newly learned materials from this course with the given checklist. With the difficulties that occurred, I was able to complete this this project over a span of eight days. At one point I had to restart from scratch I ended. For example the variables including the elements from the structure did not contain consistent information, but I ended up fixing them through trial and error along with the help of the Gaddis book and <http://www.cplusplus.com>. Ultimately, I learned and got more comfortable with new variables such as c->element and \*\*ptr.

**Flowchart**



**Pseudo Code**

*Initialize*

*If enemy1 is chosen*

*Displays status of enemy1 and player*

*If a player button is pressed*

*Player has no health*

*Exit the game*

*Else enemy1 has no health*

*Game continues to enemy1 and/or 2*

*Else if enemy2 is selected*

*Displays code to be decrypted*

*If a player button is pressed*

*Loops until code is correct*

*If code is correct displays status of enemy1 and player2*

*If player has no health*

*Exit the game*

*Else enemy1 has no health*

*Game finishes*

*Display the result*

*If a player button is pressed*

*Status writes and reads a text file*

*Else*

*Game exits*

**Major Variables**

|  |  |  |  |
| --- | --- | --- | --- |
| **Type** | **Variable** | **Description** | **Location** |
| **Integer** | \*c-> playerDamage | The value of the player’s damage | int combatBoss();  int main();  int minion(); |
|  | c-> playerHealth | The value of the player’s health | int combatBoss();  int main();  int minion(); |
|  | minionHealth | Stores the minions’ health | Minion(); |
|  | bossHealth | Stores the constantly changing value of the boss’s health | combatBoss(); |
|  | bossDamage | Stores the constantly changing value of the boss’s damage | combatBoss(); |
|  | overDose | Holds the number of potions | int minion(); |
|  | \*inputPtr | Pointer array for the encrypted input | int boss();  void decryptPrompt(); |
|  | digitInput | Stores input for decryption | Void decryptPrompt(); |
|  | inputCap (constant) | Makes sure that only four digits are taken for decryption | int boss();  void decryptPrompt();  int \*inputDecryption(); |
|  | endgameInput | Allows player to save data to a text file. | int main(); |
|  |  |  |  |
| **Char** | action | Allows player to select option at the beginning of game | int minion();  int main(); |
|  | \*namePtr | Points to the array userName | int main(); |
|  | playerInput | Allows the player to selection options during boss fight | Int combatBoss(); |
|  | userName[] | Stores the player’s username | struct Play{};  int main(); |
|  |  |  |  |
| **Bool** | btw07 | Makes sure that digitInput is between 0 and 7 digits | void decryptPrompt();  int boss(); |

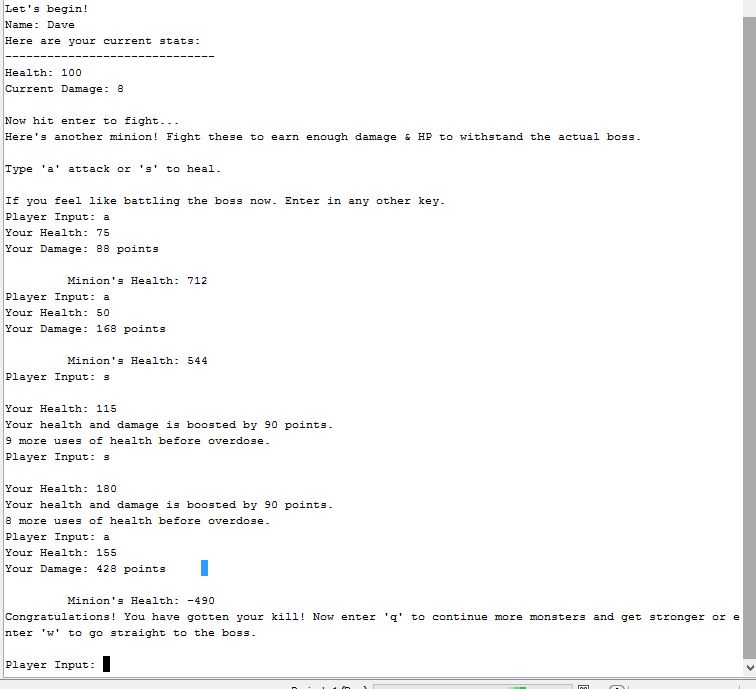
**Concepts**

From chapter 9 we learned about pointer variables and their purpose hold the memory addresses of other variables. I used this to hold the address of the player’s inputted username with in my gaming project along with holding the address for the decryption part of the game. For chapter 10 we ventured more into the string class. Practically the c-string which I used as the variable for storing the person’s name, where a for loop prints it vertically as a greeting. From chapter 11 the concepts of structures were introduced. Within my program I have a structure to hold bother integers and characters data types of the player’s current status. Lastly, chapter 12 is about advanced file operations. With this concept, I programmed it so that the player is able to store or view their most recently saved stats if they were to reach the final boss.

**Sample Input/Output**

Here is an example of the input and output of the game when fighting a minion.

Note: Even though the heal does boost health points by 90, the minion’s attack points have to be figured out and taken into consideration.



**References**

As mentioned above in the summary, at the beginning I did have difficulties with the project. Fortunately, the Gaddis book, [www.cplusplus.com](http://www.cplusplus.com), along with my previous homework assignments I was able to fix those problems. For example, through the tutorials on cplusplus.com, I was able to borrow the code for inputting and outputting contents into a text document along with help in the things I wasn’t that clear about in structures. Another example was when I look back in the Gaddis book and reminded myself that an array is a pointer so that eliminated the confusion when I wanted to user a pointer on an array.

**Checklist:**

1)Pointers

line 18: int \*playerDamage //a pointer as an element inside a structure

line 63: \*c->playerDamage

line 240 – 261 has some more examples of pointers being used

2) Functions with Structures

line 313: int combatBoss(struct Play \*c)

line 113: int minion(char action, struct Play \*c)

3) Pointers with Arrays

line 61: namePtr = c->userName;

line 280: inputPtr[count] = (digitInput % 10);

4) Array of Structures

line 20: char userName[10]; (INTERNALLY)

line 61: namePtr = c->userName; (EXTERNALLY)

5) Pointer Notation

line 67: cout<<\*(namePtr+i)<<…

6) Pointer with Structures

Line 18: int \*playerDamage; (INTERNAL)

Line 61 - 67 shows an example with pointer with structures EXTERNALLY.

**Program**

**/\***

**\* File: Project\_1.cpp**

**\* Author: Kevin Vo**

**\* Course: CSC-17A (48130)**

**\* Created on October 12, 2014, 2:11 PM**

**\*/**

**#include <cstdlib>**

**#include <iostream>**

**#include <ctime>**

**#include <iostream>**

**#include <fstream>**

**#include <cstdio>**

**#include <cstring>**

**using namespace std;**

**struct Play{**

**int \*playerDamage;**

**int playerHealth;**

**char userName[];**

**};**

**//Function Prototypes**

**int combatBoss(Play \*);//Initiates boss battle**

**//Decryptin Prompt code for to get to boss level**

**void decryptPrompt();**

**void changer(int &, int &);**

**void displayResult(int\*, string ="");**

**int \*inputDecryption(int \*);**

**int boss();//Congratulates for getting to boss level and requests code**

**int minion(char, Play \*);//Starts the minion battles for leveling up**

**//Displays instruction, story progression, and status text**

**void endBattle();**

**void duringBattle();**

**void displayStatPlayer();**

**void congratDisplay();**

**//CONSTANT global variable for in-game decrpytion**

**const int inputCap = 4;**

**int main(int argc, char\*\* argv){**

**//Allocates memory so that the members of the structure can be used**

**Play \*c = new Play;**

**int dmg = 8, endgameInput;**

**c->playerDamage = &dmg;**

**c->playerHealth = 100;**

**char action;//Input for the player to play the game**

**//Prompts the player about the game**

**cout<<"Warriors' Extravaganza is an Roleplaying Game that soley depends on "**

**<<"repetition until you understand the game.\nTo play, each turn you "**

**<<"are given a chance to fight a boss or one of its minions. Each "**

**<<"minion fight will give you a boost damage points.\n"**

**<<"Hit enter to continue...";**

**cin.ignore();**

**//uses structure member "userName" to store name**

**cout<<"Enter the name of your character: ";**

**cin.get(c->userName, 10);**

**char \*namePtr;**

**//Assigns a pointer to an element**

**namePtr = c->userName;**

**//Pointer Notation example**

**cout<<"Hello\n";**

**for(int i=0; i < 6; i++){**

**cout<<\*(namePtr+i)<<"\t\t"<<namePtr[i]<<endl;**

**}**

**cout<<"Let's begin!\n";**

**//Uses member "userName" and \*playerDamage to display string**

**cout<<"Name: "<<c->userName<<"\n"**

**<<"Here are your current stats:\n------------------------------\n"**

**<<"Health: "<<c->playerHealth<<endl<<"Current Damage: "<<**

**\*c->playerDamage<<endl<<endl<<"Now hit enter to fight...";**

**cin.ignore();**

**cin.get();**

**//Calls for minion battle**

**minion(action, c);**

**//Calls for boss battle**

**combatBoss(c);**

**//Displays congrats message**

**congratDisplay();**

**//Asks to write/read a .txt or end the game**

**cout<<"Player Input: ";**

**cin>>endgameInput;**

**if(endgameInput == 1){**

**ofstream myfile ("example.txt");**

**if (myfile.is\_open()){**

**myfile<<"\n\n---File Content---\nName: "<<c->userName<<endl;**

**myfile<<"Health: "<<c->playerHealth<<endl;**

**myfile<<"Damage: "<<\*c->playerDamage<<endl;**

**myfile.close();**

**}**

**else cout<<"Cannot open file.";}**

**else if(endgameInput == 2){**

**string line;**

**ifstream myfile ("example.txt");**

**if (myfile.is\_open()){**

**while(getline (myfile,line)){**

**cout<<line << '\n';}**

**myfile.close();}//Closes the file**

**else cout << "Cannot open file";}**

**else{cout<<"\nBonne journée!\n";}**

**delete[] namePtr;**

**return 0;}**

**//Function for duing gameplay**

**int minion(char action, struct Play \*c){**

**c-> playerHealth;**

**int minionHealth = 800;**

**c-> playerDamage;**

**int overDose = 10;**

**//Prompts player of the battle**

**duringBattle();**

**//Prompts user to push any other key to fight the boss right away.**

**cout<<"\nIf you feel like battling the boss now. Enter in any other key.\n";**

**//Loops until the player either dies or decides to fight the boss**

**do{**

**//Requests user to select the options during gameplay**

**cout<<"Player Input: ";**

**cin>>action;**

**//If players enter 's' they will get a boost in health and dmg**

**if(action == 's' || action == 'S'){**

**overDose--;**

**c->playerHealth = c->playerHealth + 90;**

**\*c->playerDamage += 90;**

**minionHealth = minionHealth - \*c->playerDamage;**

**c->playerHealth = c->playerHealth - 25;**

**cout<<"\nYour Health: "<<c->playerHealth<<endl;**

**cout<<"Your health and damage is boosted by 90 points.\n";**

**cout<<overDose<<" more uses of health before overdose.\n";**

**//If player drinks to many potions they will OD and die**

**if(overDose <=0){//If player drinks all pots they will OD and die**

**cout<<"You have died by an overdose.\n";**

**cout<<"Note: You don't get to store your stats to a .txt\n"**

**<<"until you have reach the boss.\n";**

**exit (EXIT\_FAILURE);}//Exits out the game.**

**}**

**else if(action == 'a' || 'A'){**

**\*c->playerDamage += 80;**

**minionHealth = minionHealth - \*c->playerDamage;**

**c->playerHealth = c->playerHealth - 25;**

**//Warns player of his/her low health**

**if(c->playerHealth <= (c->playerHealth/2)){**

**cout<<"\*\*\*WARNING: You have reached half health or less.\n"**

**<<"But heal too much and you will die.\n"**

**<<"Enter 's' or you will die!\n";**

**}**

**//Displays players health and current damage and minion health**

**cout<<"Your Health: "<<c->playerHealth<<endl;**

**cout<<"Your Damage: "<<\*c->playerDamage<<" points\n";**

**cout<<"\n\t Minion's Health: "<<minionHealth<<endl;**

**if (c->playerHealth <= 0){//Ends game when user dies**

**cout<<"Sorry you have die. GAMEOVER!"<<endl;**

**exit (EXIT\_FAILURE);**

**}**

**if(minionHealth <= 0){//Conditions if a minion is killed**

**c->playerHealth = c->playerHealth + 500;**

**endBattle();//Displays the options to fight boss/minions**

**//If player input w it calls for function boss**

**if(action == 'w' || action == 'W'){**

**boss();//function -> goes to the next level**

**}**

**//Occasional deduction for bonuses**

**if(action == 's' || action == 'S'){**

**overDose--;**

**c->playerHealth = c->playerHealth = 50;**

**cout<<"Your health is boosted by 50 points.\n";**

**cout<<overDose<<" more uses of health before overdose.\n";**

**//if death by OD = gameover in deduction moment of game**

**if(overDose <=0){**

**cout<<"You have died by an overdose.\n";**

**exit (EXIT\_FAILURE);}**

**}**

**else if (action == 'q' || action == 'Q'){//initi. minion rematch**

**minionHealth = 1500;//minion gets stronger**

**duringBattle();//Displays prompt during battle**

**//Warns player about their quicker lost in health**

**if(c->playerHealth <= (c->playerHealth/2)){**

**cout<<"\*\*\*WARNING: Your health is considerably low"**

**<<".\nEnter 's' or you will die!\n";**

**}**

**}**

**}**

**}**

**}while(action == 'a' || action == 'A'|| action == 's' || action == 'S' );**

**return 0;}**

**//Displays during battle instructions**

**void duringBattle(){**

**cout<<"Here's another minion! Fight these to earn enough damage & HP "**

**<<"to withstand the actual boss.\n\nType 'a' attack or 's' to heal.\n";**

**}**

**//Displays end battle options**

**void endBattle(){**

**cout<<"Congratulations! You have gotten your kill! Now "**

**<<"enter 'q' to continue more monsters and get stronger "**

**<<"or enter 'w' to go straight to the boss.\n\n";**

**}**

**//Congratulates and ask for decryption code**

**int boss(){**

**int digitInput, selectInput;**

**//pointer with array example**

**int \*inputPtr = new int [inputCap];**

**bool btw07;**

**char input;**

**cout<<"\n\n\*\*\*Congratulations you have reached the final boss but there's a "**

**<<"lock on the door.\nOh wait! You have picked up a decoder from the "**

**<<"last minion you have killed.\n";**

**cout<<"\nHmmm... there seems to be a combination the wall of 4 digits.";**

**cout<<"\nThe combination on the wall is: 1234\n\n";**

**cout<<"Please pull out the decoder by entering 'a'.\n";**

**cin.ignore();**

**decryptPrompt();//Displays and asks for combination on wall**

**delete []inputPtr;//clears memory**

**return 0;}**

**//Use to swap the digits**

**void changer(int &input, int &INPUT) {**

**int swap;**

**swap = input;**

**input = INPUT;**

**INPUT = swap;**

**}**

**//Undo the ecryption by working backwards**

**int \*inputDecryption(int \*inputDecrypting) {**

**// undo the swapping made during encryption**

**changer(\*(inputDecrypting), \*(inputDecrypting + 1));**

**changer(\*(inputDecrypting + 2), \*(inputDecrypting + 3));**

**for (int demod = 0; demod != inputCap; ++demod){**

**if (\*(inputDecrypting + demod) < 3){**

**\*(inputDecrypting + demod) = (\*(inputDecrypting + demod) + 8);**

**}**

**else{**

**\*(inputDecrypting + demod) = \*(inputDecrypting + demod);**

**}**

**//subtracts the 3 that was added during encryption**

**\*(inputDecrypting + demod) -= 3;}**

**return inputDecrypting;}**

**//Displays the encrypted or decrypted value**

**void displayResult(int \*displayer, string input2string) {**

**for (int count = 0; count != inputCap; ++count) {**

**cout<<displayer[count]<<input2string;**

**}**

**}**

**//Asks for found code**

**void decryptPrompt(){**

**int digitInput, selectInput;**

**int \*inputPtr = new int [inputCap];**

**bool btw07;**

**cout<<"Enter in the code.\n";**

**cout<<"Input Data: ";**

**cin>>digitInput;**

**do{**

**cout<<"Urgent!\n";**

**cout<<"Please re-enter '1234' (without the ' ').\n";**

**cout<<"Player Input: ";**

**cin>>digitInput;**

**}while(digitInput != 1234);**

**for (int count = 3; count >= 0; --count) {**

**inputPtr[count] = (digitInput % 10);**

**digitInput = (digitInput/10);**

**}**

**//Checks if the inputted value**

**for (int count = 0; count != 4; ++count){**

**if (\*(inputPtr + count) > 7){**

**btw07 = 0;**

**}**

**btw07 = 1;**

**}**

**if (btw07) {**

**inputPtr = inputDecryption(inputPtr);**

**cout<<"Decrypted Code: ";**

**displayResult(inputPtr);**

**}**

**delete []inputPtr;//clears memory**

**//Asks to input decrypted code**

**int decrptInput;**

**cout<<"\nNow please enter the Decrypted Code (It is 7610): ";**

**cin>>decrptInput;**

**//code input validation**

**if (decrptInput != 7610){**

**cout<<"You have inputted the wrong code...\n";**

**cout<<"Please try again... (the code is 7610)\n";**

**cout<<"User Input: ";**

**cin>>decrptInput;**

**}**

**cout<<"\nThe doors are opening. Please hit enter to continue...";**

**cin.ignore();**

**cin.get();**

**}**

**//Calculates and prompts the boss battle**

**int combatBoss(struct Play \*c){**

**int bossHealth = 1000000, bossDamage = 100;**

**char playerInput;**

**c-> playerHealth;**

**c-> playerDamage;**

**//Prompts user about boss**

**cout<<"The boss is now here. He does more damage & has way more health "**

**<<"than your previous enemies.\n"**

**<<"Attack him by hitting 'a'.\nPlayer Input: ";**

**cin>>playerInput;**

**//Loops while in boss fight**

**do{**

**//Calculations for boss' and player's status**

**bossHealth = bossHealth - c-> playerHealth;**

**c-> playerHealth = (c-> playerHealth\*3);**

**bossDamage = (bossDamage \* 10);**

**c-> playerHealth = c-> playerHealth - bossDamage;**

**cout<<"\n---Boss' Status---:\n";**

**cout<<"\tBoss' Health: "<<bossHealth<<endl;**

**cout<<"\tBoss' Damage: "<<bossDamage<<endl;**

**cout<<"\nxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx\n";**

**cout<<"---Your Status---:\n";**

**cout<<"\tYour Health: "<<c-> playerHealth<<endl;**

**cout<<"\tYour Current Damage: "<<\*c->playerDamage<<endl;**

**cout<<"Input 'a' to ATTACK and 's' to Heal-ATTACK\n";**

**cout<<"Player Input: ";**

**cin>>playerInput;**

**if(playerInput == 's'|| playerInput == 'S'){**

**bossHealth = bossHealth - \*c->playerDamage;**

**\*c->playerDamage = (\*c->playerDamage\*5);**

**c-> playerHealth = c-> playerHealth + 190;**

**//Multiplies bossDamage for every round**

**bossDamage = (bossDamage \* 2);**

**}**

**//Congrats the player for killing the boss**

**if(bossHealth <=0){**

**cout<<"\n\*\*\*Congratulations you have defeated the boss!\*\*\*.\n";**

**return 0;}**

**//Shows gameover screen if player was killed by the boss**

**if(c-> playerHealth <=0){**

**cout<<"You were killed by the boss.\nGAMEOVER.\n";**

**cout<<"Note: Kill more minions next time.\n";**

**return 0;}**

**}while(playerInput=='a'||playerInput=='A'||**

**playerInput=='s'||playerInput=='S');**

**return 0;}**

**//Displays congratulations for beating the boss**

**void congratDisplay(){**

**cout<<"Congratulations for getting to the boss win or not you are still"**

**<<"a hero!\n";**

**cout<<"Now what would you like to do?\n";**

**cout<<"Enter 1 to write your status to a text file.\n";**

**cout<<"Enter 2 to view your last status.\n";**

**cout<<"Enter 0 to exit game.\n";**

**}**