Project 2

Title

Lord of the Rings
Version 2.0

Course

CIS 17C - 48596

Due Date

December 8, 2014

Author

Brendon Quaintance

Introduction

Title: Lord of the Rings

This is a role-playing game (RPG) is a game in which the player assumes the role of a character from Lord of the Rings. The game follows the same guidelines of the actual movie: defeat the enemy and destroy the ring of power. This game allows the user to pick from 3 different heroes to fight an enemy. Each Hero has specific advantages, disadvantages. After picking a hero, the user can buy items at an item store. After that, they are sent into the battle with the enemy. While on the battle field, the user can chose to use 3 different attacks or use items that they have purchased from the Item Store. Random numbers are generated used to determine attack damages, slow attack chance, missing attacks, and picking an attack move for the enemy. Each player has a 25% chance of missing their attack and the game goes on until one of the players run out of health.

After the game is over, the user has the choice to rematch with the same hero, reselect a different hero, view the damage binary search tree, view the hash damage table, or quit the game.

Summary

Project Size: 1775 Lines of Code
Number of Variables: ~100
Number of Methods: 31

I have been working on this project for the past couple of months, as I over obsessed in making a functional game since it is the first game that I have made. I have learned a great deal of knowledge in the process of getting this game to function right. I included all the possible elements into this game that I could think of to maximize variety and make the game as fun as possible. However, this C++ version is pure text displaying and could really use a nice User Interface. On the bright side, I achieved this task when converting this game for my JavaScript class.

This project includes the following concepts that we have covered in the course:

(G) Tony Gaddis - Starting out with C++
(AD) Adam Drozdek - Data Structures and Algorithms in C++
(IA) Cormen, et.al. - Introduction to Algorithms
(BP) Bruno Priess - Data Structures and Algorithms in C++ online

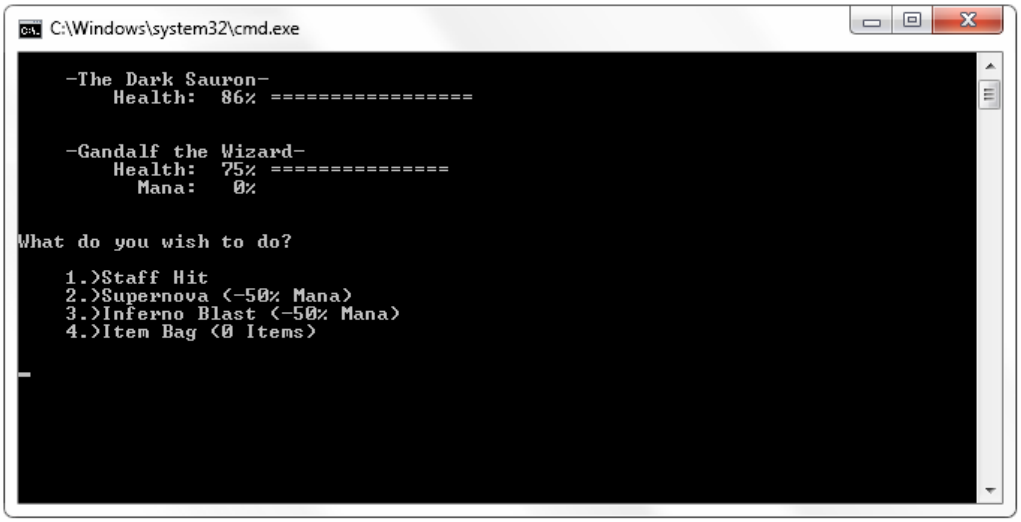
Book/Chapters	Concepts Included in Project
(AD-1) (BP-1,18) (G13,14,15,16)	• Pointers (header.h Line 48) • Inheritance (header.h Line 13) • Object Orientation (header.h Line 13)
(AD-4) (BP-6,7) (G-18)	• Stacks and Queues (main.cpp Line 400)
(AD-1)	• Standard Template Library (main.cpp Line 400)
(AD-5) (G-19) (IA-4)	• Recursion (header.h Line 107)
(AD-9) (BP-16) (G-8) (IA-6 to 9)	• Sorting (main.cpp Line 219)
(AD-10) (BP-8) (IA-11)	• Hashing (header.h Line 220)
(AD-6,7) (BP-9,10) (G-20) (IA-12to13)	• Trees (main.cpp Line 125)

Description

The main point of this program is use of Objects and their properties. Throughout the entire game, these properties are constantly being modified and changed based on the User's decisions. These properties are also being utilized by other main components of the game such as the Binary Search Tree and Hash Table.

Gameplay Visuals

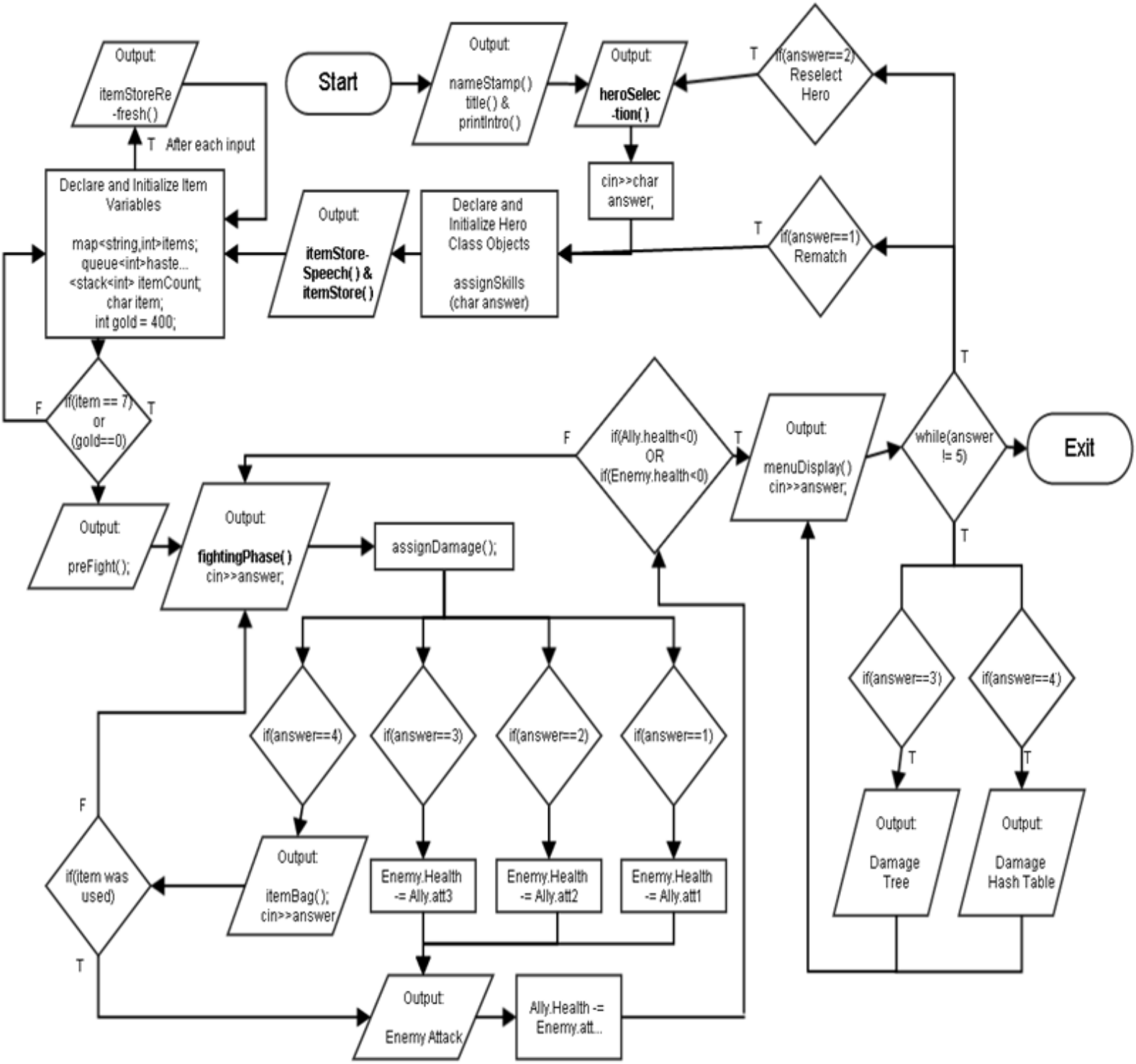
Fighting Phase:



Damage Tree:

Hash Damage Table:

Flow Chart


Start
Output:
nameStamp()
title() &
printIntro()
Output:
heroSelec
-tion( )
T
if(answer==2)
Reselect
Hero
cin>>char
answer;
T
if(answer==1)
Rematch
Declare and
Initialize Hero
Class Objects
assignSkills
(char answer)
Output:
itemStore-
Speech( ) &
itemStore( )
Declare and Initialize Item
Variables
map<string,int>items;
queue<int>haste...
<stack<int> itemCount;
char item;
int gold = 400;
T After each input
Output:
itemStoreRe
-fresh()
F
if(item == 7)
or
(gold==0)
T
Output:
preFight();
Output:
fightingPhase( )
cin>>answer;
assignDamage();
F
if(Ally.health<0)
OR
if(Enemy.health<0)
T
Output:
menuDisplay()
cin>>answer;
while(answer
!= 5)
T
Exit
T
if(answer==4)
if(answer==3)
if(answer==2)
if(answer==1)
F
if(item was
used)
T
Output:
itemBag();
cin>>answer
Enemy.Health
-= Ally.att3
Enemy.Health
-= Ally.att2
Enemy.Health
-= Ally.att1
Output:
Enemy Attack
Ally.Health -=
Enemy.att...
if(answer==3)
if(answer==4)
T
T
Output:
Damage
Tree
Output:
Damage
Hash Table

Pseudo Code

Initialize

Call nameStamp( ), title( ), printIntro( ), & heroSelection( )

```
If player enters answer
    Declare and Initialize Hero Class Objects
   Call assignSkills(char answer)

    Call itemStoreSpeech( ), & itemStore( )

    Declare and initialize Item Variables

    If item==7 or gold==0
        Call prefight( ), & fightingPhase( )
    Else
        Call itemStore( )

fightPhase( ) function
    If answer == 1
        Enemy.Health -= Ally.att1
   Else If answer == 2
        Enemy.Health -= Ally.att2
   Else If answer == 3
        Enemy.Health -= Ally.att3
   Else If answer == 4
        Call itemBag( )
            If item was used
                Move on to Enemy Attack
            Else
                Call fightingPhase( )

    Enemy Attack
        Ally.Health -= Enemy.att…

    If Ally.health<0 or Enemy.health<0
        menuDisplay( )
```

```
    Else
        Start another round; call fightingPhase( )
```

```
menuDisplay( ) function
    While answer!= 5
        If answer==1
            Rematch; Call assignSkills(char answer)
        Else if answer == 2
            Reselect Hero; Call heroSelection( )
        Else if answer == 3
            Output Damage Tree;
        Else if answer == 4
            Output Hash Damage Table; Call hashDisplay( )
        Call menuDisplay( )
    Return 0;
```

References

Tony Gaddis - Starting out with C++
Adam Drozdek - Data Structures and Algorithms in C++
Cormen, et.al. - Introduction to Algorithms
Bruno Priess - Data Structures and Algorithms in C++ online
http://www.cplusplus.com/

Program

```
//header.h
#ifndef HEADER_H
#define HEADER_H

# include <iostream>
# include <cstdlib>
using namespace std;


//========================================== HERO CLASS
================================================
//Class that assigns Ally's specific abilities
class Hero{
public:
	int health, mana,att1,att2,att2M,att3,items,armor, att3M;string
name,att1N,att2N,att3N;
	Hero(){name="";health=0;
mana=0;att1N="";att1=0;att2N="";att2=0;att2M=0;att3N="";att3=0; att3M=0,
items=0;armor=0;}
	Hero(string name,int health,int mana,string att1N,int att1,string att2N,int
att2,int att2M,string att3N,int att3,int att3M,int items, int armor)
	{
		this->health=health;this->mana=mana;this->att1=att1;this->att2=att2;this-
>att2M=att2M;this->att3=att3;this->att3M=att3M;
		this->items=items;this->att1N=att1N;this->att2N=att2N;this->att3N=att3N;
this->name=name;this->armor=armor;
	}
	~Hero(){};
};
//=========================================================================================
================

//Node Declaration
struct node
{
    int info;
    struct node *left;
    struct node *right;
}*root;

//Tree Class Declaration
class Tree
{
    public:
        void insert(node *,node *);
        void preorder(node *);
        void inorder(node *);
        void postorder(node *);
        void display(node *, int);
        Tree()
        {
            root = NULL;
        }
};
```

```
//Inserting Element into the Tree
void Tree::insert(node *tree, node *newnode)
{
    if (root == NULL)
    {
        root = new node;
        root->info = newnode->info;
        root->left = NULL;
        root->right = NULL;
        //cout<<"Root Node is Added"<<endl;
        return;
    }
    if (tree->info == newnode->info)
    {
        //cout<<"Element already in the tree"<<endl;
        return;
    }
    if (tree->info > newnode->info)
    {
        if (tree->left != NULL)
        {
            insert(tree->left, newnode);
       }
       else
       {
            tree->left = newnode;
            (tree->left)->left = NULL;
            (tree->left)->right = NULL;
            //cout<<"Node Added To Left"<<endl;
            return;
        }
    }
    else
    {
        if (tree->right != NULL)
        {
            insert(tree->right, newnode);
        }
        else
        {
            tree->right = newnode;
            (tree->right)->left = NULL;
            (tree->right)->right = NULL;
           // cout<<"Node Added To Right"<<endl;
            return;
        }
    }
}

//Pre Order Traversal
void Tree::preorder(node *ptr)
{
    if (root == NULL)
    {
        cout<<"Tree is empty"<<endl;
        return;
    }
    if (ptr != NULL)
```

```
    {
        cout<<ptr->info<<"  ";
        preorder(ptr->left);
        preorder(ptr->right);
    }
}

//In Order Traversal
void Tree::inorder(node *ptr)
{
    if (root == NULL)
    {
        cout<<"Tree is empty"<<endl;
        return;
    }
    if (ptr != NULL)
    {
        inorder(ptr->left);
        cout<<ptr->info<<"  ";
        inorder(ptr->right);
    }
}

//Postorder Traversal
void Tree::postorder(node *ptr)
{
    if (root == NULL)
    {
        cout<<"Tree is empty"<<endl;
        return;
    }
    if (ptr != NULL)
    {
        postorder(ptr->left);
        postorder(ptr->right);
        cout<<ptr->info<<"  ";
    }
}

//Display Tree Structure
void Tree::display(node *ptr, int level)
{
    int i;
    if (ptr != NULL)
    {
        display(ptr->right, level+1);
        cout<<endl;
        if (ptr == root)
            cout<<"Root->:  ";
        else
        {
            for (i = 0;i < level;i++)
                cout<<"       ";
       }
        cout<<ptr->info;
        display(ptr->left, level+1);
    }
}
```

```
//============================== HASH TABLE ==========================================
//data structure to hold id and data, our data-structure we want to use
struct data1 {
public:
        int id; //to hold a unique id for each element
        int data; //the data for each element, I used a simple int
};

class hasher {
        data1 dt[51]; //the table to hold hashed data structs
        int numel; //number of elements in table, to check if it's full
public:
        hasher();
              void clear();
        int hash(int &id);
        int rehash(int &id);
        int add(data1 &d);
        void output(char answer);
};


int hasher::hash(int &id) {
        return (id%51);

}

int hasher::rehash(int &id) {
        return ((id+1)%51);

}


hasher::hasher() {
        //create an array of data structure
        int i;

        for(i=0;i<=50;i++) {
                dt[i].id = -1; //set all ids to -1 to show they're empty
                dt[i].data = 0; //set all data values to 0, which is default
        }
        numel = 0;
}

void hasher::clear(){
        //create an array of data structure
        int i;

        for(i=0;i<=50;i++) {
                dt[i].id = -1; //set all ids to -1 to show they're empty
                dt[i].data = 0; //set all data values to 0, which is default
        }
        numel = 0;

}


int hasher::add(data1 &d) {
```

```
        if(numel < 51) {
                //table has empty places...
                int hashed = hash(d.id);
                if(hashed >= 0 && hashed <= 50 && dt[hashed].id == -1) {
                        //slot is empty, assign new data
                        dt[hashed].id = d.id;
                        dt[hashed].data = d.data;
                        return 0;
                } else {
                       //need to rehash the id
            int i=0;

            //try every place in table to find an empty place
            while(i<=50) {
                            hashed = rehash(hashed);
                            if(dt[hashed].id == -1) {
                                    dt[hashed].id = d.id;
                                    dt[hashed].data = d.data;
                                    return 0;
                            } else if(i==50) {
                                    //couldn't find the empty place
                                    return -1; //terminate function with error
                            }
                i++; //update value of i
                                    }
             }

        } else {
                //table is full
                return (-1);
        }
}

void hasher::output(char answer) {
       /*Weak Hits  (0-49 Damage)
       Average Hits (50-99 Damage)
       Good Hits   (100-149 Damage)
       Strong Hits (150-199 Damage)
       Beast Hits   (200+ Damage)*/

       //DISPLAYS SPECIFIC RANGES IN HASH TABLE
       if(answer=='1'){
                            cout<<"\nWeak Hits (0-49 Damage):\n";
                            for(int i=0;i<10;i++) {
                                   if(dt[i].id != -1){
                                                cout<<"     "<<dt[i].data<<endl;
                                   }
                            }
                    }else if(answer=='2'){
                            cout<<"\nAverage Hits (50-99 Damage):\n";
                            for(int i=10;i<20;i++) {
                                   if(dt[i].id != -1){
                                                cout<<"     "<<dt[i].data<<endl;
                                   }
                            }
                    }else if(answer=='3'){
                            cout<<"\nGood Hits (100-149 Damage):\n";
                            for(int i=20;i<30;i++) {
```

```
                        if(dt[i].id != -1){
                                cout<<"     "<<dt[i].data<<endl;
                        }
                }
        }else if(answer=='4'){
                cout<<"\nStrong Hits (150-199 Damage):\n";
                for(int i=30;i<40;i++) {
                        if(dt[i].id != -1){
                                cout<<"     "<<dt[i].data<<endl;
                        }
                }
        }else if(answer=='5'){
                cout<<"\nBeast Hits (200+ Damage):\n";
                for(int i=40;i<50;i++) {
                        if(dt[i].id != -1){
                                cout<<"     "<<dt[i].data<<endl;
                        }
                }

        }else if(answer=='6'){
                //DISPLAYS ENTIRE HASH TABLE
                cout<<"\nAll Hits:\n";
                cout<<"\nWeak Hits (0-49 Damage):\n";
                for(int i=0;i<10;i++) {
                        if(dt[i].id != -1){
                                cout<<"     "<<dt[i].data<<endl;
                        }
                }
                cout<<"\nAverage Hits (50-99 Damage):\n";
                for(int i=10;i<20;i++) {
                        if(dt[i].id != -1){
                                cout<<"     "<<dt[i].data<<endl;
                        }
                }
                cout<<"\nGood Hits (100-149 Damage):\n";
                for(int i=20;i<30;i++) {
                        if(dt[i].id != -1){
                                cout<<"     "<<dt[i].data<<endl;
                        }
                }
                cout<<"\nStrong Hits (150-199 Damage):\n";
                for(int i=30;i<40;i++) {
                        if(dt[i].id != -1){
                                cout<<"     "<<dt[i].data<<endl;
                        }
                }
                cout<<"\nBeast Hits (200+ Damage):\n";
                for(int i=40;i<50;i++) {
                        if(dt[i].id != -1){
                                cout<<"     "<<dt[i].data<<endl;
                        }
                }
        }
        cout<<"\n\nPress any key to go back . . .";

    cout<<"\n\n_____
_____\n\n";
        cin>>answer;
```

```
}
#endif


//main.cpp
/*================================ Lord of the Rings Game
=========================================

       Author: Brendon Quaintance
       Class:  CIS-17C Professor Lehr
       Date:   December, 2014

       Lines of code: 1775
================================================================================================
=======*/

//Include Libraries
#include <iostream>
#include <string>
#include <ctime>
#include <iomanip>
#include <Windows.h>
#include "header.h"
using namespace std;

//Required Libraries for Midterm
#include <map>
#include <queue>
#include <stack>
#include <set>
#include <algorithm>


//============== Global Variables ========================================
//Tree variables
Tree bst;
node *temp;

//Hash Variables
hasher h1;
data1 d;

map<string,int>items;//sorts and displays items container with their values
queue<int> doubleDamage,invisibility,regeneration,haste;//used for pushing/each amount of
different items
stack<int> itemCount;//used for pushing/popping total item amount
int speech = 1;
//========================================================================

//Included Functions
void hashDisplay(char hero, int&speech);
void hashDamage(int);
void menuDisplay(char hero,int&speech);
void nameStamp();
void title();
void printIntro();
char heroSelection();
void assignSkills(char hero,int&speech);
```

```
void itemStoreSpeech(Hero&Ally,Hero&Enemy,int&speech);
void itemStore(Hero&Ally,Hero&Enemy);
void itemStoreRefresh(Hero&Ally,int gold);
void preFight(Hero&Ally,Hero&Enemy);
void fightingPhase(Hero&Ally,Hero&Enemy);
void getAnswer(Hero&Ally,Hero&Enemy,char&answer);
void attack(Hero&Ally,Hero&Enemy,char answer);
void refreshScreen(Hero&Ally,Hero&Enemy);
void assignDamage(Hero&Ally,Hero&Enemy);
bool itemBag(Hero&Ally,Hero&Enemy);
void dots();

/*================================== Main ==============================================
      Purpose: Main function is connected to all the other functions. After all other
                     functions are called(game is over), it asks the user if they want
to
                     rematch, reselect hero, or quit.
=======================================================================================
*/
int main(){
      char hero='a';

    nameStamp();
      //Title Page
      title();
      //Print Introduction
      printIntro();

      //Select a Hero
      hero = heroSelection();

      //Assigning Hero properties And Begin Fight
      assignSkills(hero,speech);

      menuDisplay(hero,speech);

      return 0;
}
void menuDisplay(char hero, int&speech){
      system("CLS");

      //used for sorting and displaying Main Menu Screen
      set<string> mainMenu;
      mainMenu.insert("              2.)Reselect Hero");
      mainMenu.insert("              5.)Quit");
      mainMenu.insert("              1.)Rematch");
      mainMenu.insert("              3.)View Damage Tree");
      mainMenu.insert("              4.)View Damage Hash Table");

   char choice='A';
      cout<<"\n\n_____________________________________ MAIN MENU
_____________________________________\n\n";
            for (set<string>::const_iterator p = mainMenu.begin( );p != mainMenu.end(
); ++p){
                     cout << *p << endl;

      }cout<<"\n_____________________________________________________________________
_______";
```

```
		cout<<endl<<endl;
		cin>>choice;

	while((choice!='1')&&(choice!='2')&&(choice!='3')&&(choice!='4')&&(choice!='5')){
			system("CLS");
			cout<<"\n\n_____ MAIN MENU
_____\n\n";
			for (set<string>::const_iterator p = mainMenu.begin( );p !=
mainMenu.end( ); ++p){
				cout << *p << endl;
			}

	cout<<"\n_____
_____";
			cout<<"\n\nInvalid choice!"<<endl<<endl;
			cin>>choice;
		}
		if(choice == '1'){
			assignSkills(hero,speech);
		}
		if(choice == '2'){
			hero = heroSelection();
		}
		if(choice == '3'){
			char back;
			system("CLS");
			cout<<"\n\n_____ DAMAGE TREE
_____\n";
			cout<<" -Displaying the Tree that consists of all damage done to the
enemy."<<endl<<endl;
			 cout<<"Inorder Traversal of Tree:"<<endl;
	    bst.inorder(root);
		cout<<endl;

		cout<<"Preorder Traversal of Tree:"<<endl;
		bst.preorder(root);
		cout<<endl;

		cout<<"Postorder Traversal of Tree:"<<endl;
		bst.postorder(root);
			cout<<"\n\nTree Visual:";

		bst.display(root,1);
		cout<<"\n\nPress any key to go back . . .";

	cout<<"\n_____
_____";
			cin>>back;
			menuDisplay(hero,speech);
		}

		if(choice == '4'){
			hashDisplay(hero,speech);
		}

		if(choice == '5'){
			exit(1);
		}
```

```
		menuDisplay(hero,speech);

}
void hashDamage(int dmg){
	d.data = dmg;

	//Assign Damage to Hash Table
	if((0<d.data)&&(d.data<50)){
		d.id=1;
	}else if((50<=d.data)&&(d.data<100)){
		d.id=10;
	}else if((100<=d.data)&&(d.data<150)){
		d.id=20;
	}else if((150<=d.data)&&(d.data<200)){
		d.id=30;
	}else if((200<=d.data)){
		d.id=40;
	}
	 h1.add(d);
}
void hashDisplay(char hero,int&speech){
	char answer;
			system("CLS");
			cout<<"\n\n_____ DAMAGE HASH
_____\n";
			cout<<" -Which Hash Range would you like to display?\n\n"
			 "		1.)Weak Hits     (0-49 Damage)\n"
			 "		2.)Average Hits (50-99 Damage)\n"
			 "		3.)Good Hits   (100-149 Damage)\n"
			 "		4.)Strong Hits (150-199 Damage)\n"
			 "		5.)Beast Hits   (200+ Damage)\n"
			 "		6.)All Hits\n"
			 "		7.)Back\n\n";

	cout<<"\n_____
_____\n";

			cin >> answer;


	while((answer!='1')&&(answer!='2')&&(answer!='3')&&(answer!='4')&&(answer!='5')&&(
answer!='6')&&(answer!='7')){
				cout<<"Invalid Choice!";
				cin>>answer;
			}

			if(answer=='7'){
				menuDisplay(hero,speech);
			}
			system("CLS");

			cout<<"\n\n_____ DAMAGE HASH
_____\n";
			h1.output(answer);

			hashDisplay(hero,speech);

}
```

```
/*================================== Name Stamp ========================================
       Purpose: Displays my name, date, class, and project when opened
==========================================================================================
*/
void nameStamp(){
       cout<<"\n\n\n\n\n\n\n\n\n\n                                ";
       Sleep(500);
       string s1("Brendon Quaintance\n\n\n                              ");
       string s2("CIS-17C Final Project\n                                 ");
       string s3("Professor Lehr");
       string s4("December, 2014\n                                    ");

       string array [4] = {s3,s1,s4,s2};

       //Sort String Array
       sort(array,array+4);

       for (int i = 0; i < 4; i++){
              string temp = array[i];
              for(int j =0; j<temp.size();j++){
                     cout << temp[j];
                     Sleep(15);
              }
              Sleep(500);
       }
       Sleep(2000);
       system("CLS");
}
/*================================== Title =====================================
       Purpose: Prints out the title "Lord of the Rings" letter by letter.
==============================================================================
*/
void title(){
       cout<<"\n_____________________________________________________________________________
_______";
         string t1("  _\n");
         string t2(" | |                                  __  ___\n");
         string t3(" | |                                /  \\| __|\n");
         string t4(" | |       ____  ____    __         |||||_|      ____    _  _    _
_____  __\n");
         string t5(" | |      /    \\|    \\  |   \\      \\__/|_|      |    \\  | || \\
| |/  _  \\/  \\\n");
         string t6(" | |      | || || ||  | | || \\ ____  _   _  ___ | ||  | | ||  \\ |
|| / |_|||\\_|\n");
         string t7(" | |      | || ||    /  | || ||_  _|| |_| || __||    /  | ||   \\| ||
| __ ||_  \n");
         string t8(" | |      | || || |\\ \\  | || |  | | |  _  || _| | |\\ \\  | || |\\
|| ||_ |\\__\\ \n");
         string t9(" | |_____ | || || | \\ \\ | || /  |_| |_| |_||___|| | \\ \\ | || | \\
|| |_/ /__| |\n");
        string t10(" |_______|\\____/|_|  \\_\\|___/                  |_|  \\_\\|_||_|
\\_|\\____/ \\___/ \n");
       string array [10] = {t1,t2,t3,t4,t5,t6,t7,t8,t9,t10};

       for (int i = 0; i < 10; i++){
              string temp = array[i];
              for(int j =0; j<temp.size();j++){
                     cout << temp[j];
```

```
                        Sleep(10);
                }
        }
        cout<<"_____
_____";
        cout<<"\n                          ";
        Sleep(3000);

        string name("By: Brendon Quaintance");
        for(int i = 0;i<name.size();i++){
                cout<<name[i];
                Sleep(30);
        }
        Sleep(5000);
        cout<<"\n\n";
}
/*================================= Print Introdcution
=========================================
        Purpose: This function Prints out the introduction of letter by letter
================================================================================================
*/
void printIntro(){
        cout << "\n\n    ";

        string intro("      After hiding and lying dormant for 500 years, Sauron, an Evil
Dark\n\n Lord, started to reveal himself during the Second Age.");
        string intro2(" By the year of SA\n\n 1000 he had grown powerful enough to
establish himself in Mordor in eastern\n\n Middle-earth, where he started building the
Dark Tower of Barad-dur close to\n\n Mount Doom.");
        string intro3(" Sauron began assembling armies of Uruk, Graugs and other
creatures,\n and started to corrupt the hearts of Men, giving them delusions of wealth
and\n\n power.");
        string intro4(" In the Second Age, Sauron took on a fair form and used it under
the\n\n alias Annatar, or the Lord of Gifts, to deceive the Elves into creating the\n\n
One Ring.");
        string intro5(" While gaining enormous amounts of power with the One ring, he
began\n\n to rule all of Mordor and enslave all Allies.");
        string intro6(" All hope of survivability is up\n\n to you. You must destroy him
and the Ring of Power!");

        string array [6] = {intro,intro2,intro3,intro4,intro5,intro6};


        for (int i = 0; i < 6; i++){
                string temp = array[i];
                for(int j =0; j<temp.size();j++){
                        cout << temp[j];
                        Sleep(15);
                }
                Sleep(3500);
        }
        dots();
        Sleep(1000);
}
/*================================= Hero Selection
=========================================
        Purpose: This function prompts the user the 3 different heroes and their
advantages
```

```
                    and disadvantages. The user then picks one of the three heros
================================================================================================
*/
char heroSelection(){
	system("CLS");
	char answer;

	cout<<"HERO
SELECTION:\n_____________________________________________________________________________
__\n"
		<<"     ========)==================== Aragorn
==============================>\n          Aragorn's skills in battle lay primarily in
his sword craft. He"
		<<"\n     is a mighty warrior with the sword and easily defeated many types
of\n     foes, suchas Trolls and Ringwraiths.\n"
		<<"               Advantage:    Has Armor That Absorbs Extra Damage\n
Disadvantage: Slower Attacks"
		<<"\n\n     >>>========================== Legolas
==============================>\n          Legolas is an elf of the Woodland Realm of
Mirkwood. Legolas \n"
		<<"     specializes in using a bow against the wolves who attack the
Fellowship\n     in the mountain pass and against goblins in the Fight in Balin's Tomb."
		<<"\n               Advantage:    Can Fire Attacks Multiple Times\n
Disadvantage: Lower Damage Output"
		<<"\n
__"
		<<"\n     ============================== Gandalf
============================(__)\n          Gandalf is wizard of the Secret Fire, wielder
of the flame of\n"
		<<"     Anor, and bearer of Narya. He creates blinding flashes and other\n
pyrotechnics to distract the goblins of the Misty Mountains."
		<<"\n               Advantage:    High Magic Attacks\n
Disadvantage: Consumes High Amounts of
Mana\n_____________________________________________________________________________";

	cout<<"\nWhich Hero Would You Like to Play?  1.Aragon  2.Legolas  3.Gandalf\n";
	cin>>answer;

	while((answer!='1')&&(answer!='2')&&(answer!='3')){
		cout<<"Not a valid choice!\n\n";
		cin>>answer;
	}
	return answer;
}
/*================================ Assign Skills
=======================================
	Purpose: This function assigns property values base on which hero they chose. It
then
		sends the user to the items store(), prefight(), fightingPhase(),
and
		deconstructs the objects after the fighting phase is over.
================================================================================================
*/
void assignSkills(char hero,int&speech){
	//Clears Hash Table
	h1.clear();
```

```
	Hero Enemy("The Dark Sauron",800,0,"Mind Decay",0,"Black Toxic Fog",0,0,"Smack
Down",0,0,1,0);
	if(hero=='1'){
		Hero Ally("Aragorn the Swordsman",200,0,"Spinning Slash",0,"Dagger
Stab",0,0,"Lunge Attack",0,0,0,200);
			itemStoreSpeech(Ally,Enemy,speech);
			preFight(Ally,Enemy);//Prefight Prompt
			fightingPhase(Ally,Enemy);//Send Hero to Finght
			Ally.~Ally();//Delete Object after fight
	}else if(hero=='2'){
		Hero Ally("Legolas the Archer",200,100,"Arrowhead Assault",0,"Agility
Arrow",0,25,"Focused Shot",0,0,0,0);
			itemStoreSpeech(Ally,Enemy,speech);
			preFight(Ally,Enemy);//Prefight Prompt
			fightingPhase(Ally,Enemy);//Send Hero to Fight
			Ally.~Ally();//Delete Object after fight
	}else if(hero=='3'){
		Hero Ally("Gandalf the Wizard",200,100,"Staff
Hit",0,"Supernova",0,50,"Inferno Blast",0,50,0,0);
			itemStoreSpeech(Ally,Enemy,speech);
			preFight(Ally,Enemy);//Prefight Prompt
			fightingPhase(Ally,Enemy);//Send Hero to Fight
			Ally.~Ally();//Delete Object after fight
	}
	Enemy.~Enemy();
}
void itemStoreSpeech(Hero&Ally,Hero&Enemy,int&speech){
	system("CLS");

	if(speech==1){
		string talk("\n\n\n\n\n\n\n\n\n     Before leaving for battle, you have Gold
Coins that can be used to\n\n     purchase items that can assist in you defeating the
Evil Dark Lord");
		for (int i = 0; i < talk.size(); i++){
			cout << talk[i];
			Sleep(15);
		}
		dots();
		string talk2("\n\n\n\n              Let's visit Frodo at the Item Store in
Hobbiton");
		for (int i = 0; i < talk2.size(); i++){
			cout << talk2[i];
			Sleep(15);
		}
		dots();
	}
	itemStore(Ally,Enemy);
	speech=0;
}
/*============================== Item Store
==========================================
	Purpose: This function displays the items that are in the store for the user to
purchase.
=============================================================================================
=*/
void itemStore(Hero&Ally,Hero&Enemy){
	char item= 'a';
	char ans = 'a';
```

```
	int gold = 400;
	string item1="Health Potion";int item1C=100;
	string item2="Mana Potion";int item2C=100;
	string item3="Double Damage";int item3C=100;
	string item4="Invisibility";int item4C=100;
	string item5="Regeneration";int item5C=300;
	string item6="Haste";int item6C=100;

	//======== Start with no Items ===============
	items["5.)Invisibility "]=0;
	items["3.)Regeneration "]=0;
	items["1.)Health Potion"]=0;
	items["2.)Mana Potion  "]=0;
	items["4.)Double Damage"]=0;
	items["6.)Haste        "]=0;

	//======== Delete Any Existing Active Items ===============
	while
((!itemCount.empty())&&(!doubleDamage.empty())&&(!invisibility.empty())&&(!regeneration.e
mpty())&&(!haste.empty()))
	 {
		doubleDamage.pop();
		invisibility.pop();
		regeneration.pop();
		haste.pop();
	 }
	while(!itemCount.empty()){
		itemCount.pop();
	}

	string name;
	system("CLS");
	cout<<"\n_____________________________________ Item Store
_____________________________________\n";

	string talk("      Frodo: Hello!! Welcome ");
	for (int i = 0; i < talk.size(); i++){
		cout << talk[i];
		Sleep(15);
	}
	if(Ally.name=="Aragorn the Swordsman"){
		string name("Aragorn!");
		for (int i = 0; i < name.size(); i++){
		cout << name[i];
		Sleep(15);
		}
	}else if(Ally.name=="Legolas the Archer"){
		string name("Legolas!");
		for (int i = 0; i < name.size(); i++){
		cout << name[i];
		Sleep(15);
		}
	}else{
		string name("Gandalf!");
		for (int i = 0; i < name.size(); i++){
		cout << name[i];
		Sleep(15);
		}
```

```
	}Sleep(1000);
	string talk2(" Please let me know if I can\n\n            help you find anything
you are looking for");
	for (int i = 0; i < talk2.size(); i++){
		cout << talk2[i];
		Sleep(15);
	}
	dots();
	Sleep(500);

	while(ans!='1'){
		int itemAmount = 0;

		itemStoreRefresh(Ally,gold);
		cout<<"\nFrodo: Enter In the Item Number You Wish to Purchase: ";
		cin>>item;


	while((item!='1')&&(item!='2')&&(item!='3')&&(item!='4')&&(item!='5')&&(item!='6')
&&(item!='7')){
			itemStoreRefresh(Ally,gold);
			cout<<"\nFrodo: Invalid choice!\n\n";
			cin>>item;
		}
		if(item=='1'){
			itemStoreRefresh(Ally,gold);
			cout<<"\nFrodo: How many "<<item1<<"(s) would you like to purchase?
";
			cin>>itemAmount;
			if(gold<item1C){
				itemStoreRefresh(Ally,gold);
				cout<<"\nFrodo: You do not have enough Gold!\n\n";
				Sleep(2000);
			}else if(gold<item1C*itemAmount){
				itemStoreRefresh(Ally,gold);
				cout<<"\nFrodo: You do not have enough Gold!\n\n";
				Sleep(2000);
			}else if(itemAmount>0){
				items["1.)Health Potion"]+=itemAmount;
				for(int i = 0;i<itemAmount;i++){
					itemCount.push(1);
				}
				gold -= item1C*itemAmount;
				cout<<"\nFrodo: Here is the "<<itemAmount<<" "<<item1<<"(s)
that you requested! -"<<item1C*itemAmount<<" Gold";
				Sleep(3000);
			}
		}
		if((item=='2')&&(Ally.name!="Aragorn the Swordsman")){
			itemStoreRefresh(Ally,gold);
			cout<<"\nFrodo: How many "<<item2<<"(s) would you like to purchase?
";
			cin>>itemAmount;
			if(gold<item2C){
				itemStoreRefresh(Ally,gold);
				cout<<"\nFrodo: You do not have enough Gold!\n\n";
				Sleep(2000);
			}else if(gold<item2C*itemAmount){
```

```
                        itemStoreRefresh(Ally,gold);
                        cout<<"\nFrodo: You do not have enough Gold!\n\n";
                        Sleep(2000);
                }else if(itemAmount>0){
                        items["2.)Mana Potion  "] += itemAmount;
                        for(int i = 0;i<itemAmount;i++){
                                itemCount.push(1);
                        }
                        gold -= item2C*itemAmount;
                        cout<<"\nFrodo: Here is the "<<itemAmount<<" "<<item2<<"(s)
that you requested! -"<<item2C*itemAmount<<" Gold";
                        Sleep(3000);
                }
        }else if(item=='2'){
                itemStoreRefresh(Ally,gold);
                cout<<"\nFrodo: You have no use for Mana!";
                Sleep(3000);
        }
        if(item=='3'){
                itemStoreRefresh(Ally,gold);
                cout<<"\nFrodo: How many "<<item5<<"(s) would you like to purchase?
";
                cin>>itemAmount;
                if(gold<item4C){
                        itemStoreRefresh(Ally,gold);
                        cout<<"\nFrodo: You do not have enough Gold!\n\n";
                        Sleep(2000);
                }else if(gold<item5C*itemAmount){
                        itemStoreRefresh(Ally,gold);
                        cout<<"\nFrodo: You do not have enough Gold!\n\n";
                        Sleep(2000);
                }else if(itemAmount>0){
                        items["3.)Regeneration "]+=itemAmount;
                        for(int i = 0;i<itemAmount;i++){
                                itemCount.push(1);
                        }
                        gold -= item5C*itemAmount;;
                        cout<<"\nFrodo: Here is the "<<itemAmount<<" "<<item5<<"(s)
that you requested! -"<<item5C*itemAmount<<" Gold";
                        Sleep(3000);
                }
        }
        if(item=='4'){
                itemStoreRefresh(Ally,gold);
                cout<<"\nFrodo: How many "<<item3<<"(s) would you like to purchase?
";
                cin>>itemAmount;
                if(gold<item3C){
                        itemStoreRefresh(Ally,gold);
                        cout<<"\nFrodo: You do not have enough Gold!\n\n";
                        Sleep(2000);
                }else if(gold<item3C*itemAmount){
                        itemStoreRefresh(Ally,gold);
                        cout<<"\nFrodo: You do not have enough Gold!\n\n";
                        Sleep(2000);
                }else if(itemAmount>0){
                        items["4.)Double Damage"]+=itemAmount;
                        for(int i = 0;i<itemAmount;i++){
```

```
                                        itemCount.push(1);
                                }
                                gold -= item3C*itemAmount;
                                cout<<"\nFrodo: Here is the "<<itemAmount<<" "<<item3<<"(s)
that you requested! -"<<item3C*itemAmount<<" Gold";
                                Sleep(3000);
                        }
                }
                if(item=='5'){
                        itemStoreRefresh(Ally,gold);
                        cout<<"\nFrodo: How many "<<item4<<"(s) would you like to purchase?
";
                        cin>>itemAmount;
                        if(gold<item4C){
                                itemStoreRefresh(Ally,gold);
                                cout<<"\nFrodo: You do not have enough Gold!\n\n";
                                Sleep(2000);
                        }else if(gold<item4C*itemAmount){
                                itemStoreRefresh(Ally,gold);
                                cout<<"\nFrodo: You do not have enough Gold!\n\n";
                                Sleep(2000);
                        }else if(itemAmount>0){
                                items["5.)Invisibility "]+=itemAmount;
                                for(int i = 0;i<itemAmount;i++){
                                        itemCount.push(1);
                                }
                                gold -= item4C*itemAmount;;
                                cout<<"\nFrodo: Here is the "<<itemAmount<<" "<<item4<<"(s)
that you requested! -"<<item4C*itemAmount<<" Gold";
                                Sleep(3000);
                        }
                }
                if(item=='6'){
                        itemStoreRefresh(Ally,gold);
                        cout<<"\nFrodo: How many "<<item6<<"(s) would you like to purchase?
";
                        cin>>itemAmount;
                        if(gold<item4C){
                                itemStoreRefresh(Ally,gold);
                                cout<<"\nFrodo: You do not have enough Gold!\n\n";
                                Sleep(2000);
                        }else if(gold<item6C*itemAmount){
                                itemStoreRefresh(Ally,gold);
                                cout<<"\nFrodo: You do not have enough Gold!\n\n";
                                Sleep(2000);
                        }else if(itemAmount>0){
                                items["6.)Haste        "]+=itemAmount;
                                for(int i = 0;i<itemAmount;i++){
                                        itemCount.push(1);
                                }
                                gold -= item6C*itemAmount;;
                                cout<<"\nFrodo: Here is the "<<itemAmount<<" "<<item6<<"(s)
that you requested! -"<<item6C*itemAmount<<" Gold";
                                Sleep(3000);
                        }
                }
                if(item=='7'){
                        itemStoreRefresh(Ally,gold);
```

```
                    cout<<"\nFrodo: Are You Sure You're Ready to Begin Your Journey? You
still have "<<gold<<" Gold\n       left..  1.Yes 2.No\n\n";
                    cin>>ans;
             }
             while((gold<100)&&(ans!='1')){
                    itemStoreRefresh(Ally,gold);
                    cout<<"\nFrodo: You Have Run Out of Gold!\n\nAre You Ready to Begin
Your Journey 1.Yes 2.No?\n\n";
                    cin>>ans;
             }
      }
      itemStoreRefresh(Ally,gold);
      string ally = Ally.name;
      string enemy = Enemy.name;
      string talk4("\nFrodo: Goodluck ");
      string talk5("! May ");
      string talk6(" and the Ring of \n\n       Power be destroyed!!");
      string array[5] ={talk4,ally,talk5,enemy,talk6};

      for (int i = 0; i < 5; i++){
             string temp = array[i];
             for(int j =0; j<temp.size();j++){
                    cout << temp[j];
                    Sleep(15);
             }
      }
      Sleep(3000);
}
/*============================== Item Store Refresh
=====================================
      Purpose: This function refreshes the screen in the item store in order to display
                and confirm the purchasing of items and the change of gold amount
================================================================================================
*/
void itemStoreRefresh(Hero&Ally,int gold){
      string item1="Health Potion";int item1C=100;
      string item2="Mana Potion";int item2C=100;
      string item3="Double Damage";int item3C=100;
      string item4="Invisibility";int item4C=100;
      string item5="Regeneration";int item5C=300;
      string item6="Haste";int item6C=100;
      string name="";

      if(Ally.name=="Aragorn the Swordsman"){
             name="Aragorn!";
      }else if(Ally.name=="Legolas the Archer"){
             name="Legolas!";
      }else{
             name="Gandalf!";
      }
      system("CLS");
             cout<<"\n_____________________________________ Item Store
_____________________________________\n"
                    <<"      Frodo: Hello!! Welcome "<<name<<" Please let me know if I
can\n\n             help you find anything you are looking for..."
                    <<"\n\n               Items:          Costs:
Description:\n\n"
```

```
                    <<"           1.) "<<item1<<" "<<item1C<<" Gold   -Fully Restores
Health\n"
                    <<"           2.) "<<item2<<"   "<<item2C<<" Gold   -Fully Restores
Mana\n"
                    <<"           3.) "<<item5<<"  "<<item5C<<" Gold   -Fully Restores
Health & Mana\n"
                    <<"           4.) "<<item3<<" "<<item3C<<" Gold   -Doubles Your
Attack Damage\n"
                    <<"           5.) "<<item4<<"  "<<item4C<<" Gold   -Causes Enemy's
Attack to Miss\n"
                    <<"           6.) "<<item6<<"         "<<item6C<<" Gold   -
Increases Your Attack Speed\n"
                    <<"           7.) Exit\n"

      <<"_____________________________________________________________________________
_\n"
                    <<"                              Gold Remaining: "<<gold<<"\n";
}
void preFight(Hero&Ally,Hero&Enemy){
      system("CLS");
      cout<<"\n\n\n\n\n\n\n\n\n\n\n    "<<Ally.name<<" has gained enough courage to fight
"<<Enemy.name<<"!\n\n\n\n\n\n                          ";
      Sleep(2000);
      string begin("The Battle Will Begin In: 3");
      for (int i = 0; i < begin.size(); i++){
            cout << begin[i];
            Sleep(15);
      }
      int num=3;
      Sleep(1250);
      for(int j=2; j>0;j--){
            system("CLS");
            cout<<"\n\n\n\n\n\n\n\n\n\n\n    "<<Ally.name<<" has gained enough courage to
fight "<<Enemy.name<<"!\n\n\n\n\n\n         ";
            cout << "             The Battle Will Begin In: ";
            cout<<j;
            Sleep(1250);
      }
}
/*============================== Fighting Phase
=======================================
      Purpose: This is a recursive function beings the fighting phase of the game. After
each
                  round, it checks to see if both character are still alive. If so,
it starts
                  another round by calling its own function name;
================================================================================================
*/
void fightingPhase(Hero&Ally,Hero&Enemy){
      system("CLS");
      //Receives command from user
      char answer;

      //used for generating random number
      srand(unsigned(time(0)));

      //Sets 66% Chance of Hitting
      int enemyHit = rand()%4;
```

```
//Assigns Random Damage For Each Move
assignDamage(Ally,Enemy);
//Assigns random interger from 1-9 to determine what attack the enemy will use
int enemyAttack = rand()%8+1;

refreshScreen(Ally,Enemy);
getAnswer(Ally,Enemy,answer);

if(answer=='4'){
	refreshScreen(Ally,Enemy);
	itemBag(Ally,Enemy);
}
if(answer!='4'){

	//=========== Ally Attack ================================
	attack(Ally,Enemy,answer);
}

//===== Determine if enemy is still alive ============
 if(Enemy.health<=0){
	cout<<"\n\nYou Have Slain "<<Enemy.name<<"!\n\n\n           ";
	Sleep(3000);
	string def("VICTORY!");
	for(int i=0;i<def.size();i++){
		cout<<def[i];
	}Sleep(3000);
	system("CLS");
	return;
}

//=========== Enemy Attack ================================
refreshScreen(Ally,Enemy);

if(invisibility.size()>1){
	enemyHit=3;
}

cout<<Enemy.name<<" Attacks with ";
if(enemyAttack<3){
	cout<<Enemy.att1N;
}
else if(enemyAttack<7){
	cout<<Enemy.att2N;
}
else if(enemyAttack>=7){
	cout<< Enemy.att3N;
}
dots();

//If the enemy's attack hit
if(enemyHit<3){
	if(enemyAttack<4){
		if(Ally.name=="Aragorn the Swordsman"){
			if(Ally.armor>=Enemy.att1){
				Ally.armor -= Enemy.att1;
			}else if(Ally.armor<Enemy.att1){
				Ally.armor -= Enemy.att1;
```

```
                        for(int i = 0;i>Ally.armor;i--){
                                Ally.health--;
                        }
                        Ally.armor=0;
                    }else{
                    Ally.health -= Enemy.att1;
                    }
                }else{
                        Ally.health -= Enemy.att1;
                }
                refreshScreen(Ally,Enemy);
                cout<<"His attack Hit! Damge Dealt: "<<Enemy.att1<<"\n\n";
                Sleep(3000);
        }else if(enemyAttack<7){
                if(Ally.name=="Aragorn the Swordsman"){
                        if(Ally.armor>=Enemy.att2){
                                Ally.armor -= Enemy.att2;
                        }else if(Ally.armor<Enemy.att2){
                                Ally.armor -= Enemy.att2;
                                for(int i = 0;i>Ally.armor;i--){
                                        Ally.health--;
                                }
                                Ally.armor=0;
                        }else{
                        Ally.health -= Enemy.att2;
                        }
                }else{
                        Ally.health -= Enemy.att2;
                }
                refreshScreen(Ally,Enemy);
                cout<<"His attack Hit! Damge Dealt: "<<Enemy.att2<<"\n\n";
                Sleep(3000);
        }else if(enemyAttack>=7){
                if(Ally.name=="Aragorn the Swordsman"){
                        if(Ally.armor>=Enemy.att3){
                                Ally.armor -= Enemy.att3;
                        }else if(Ally.armor<Enemy.att3){
                                Ally.armor -= Enemy.att3;
                                for(int i = 0;i>Ally.armor;i--){
                                        Ally.health--;
                                }
                                Ally.armor=0;
                        }else{
                        Ally.health -= Enemy.att3;
                        }
                }else{
                        Ally.health -= Enemy.att3;
                }
                refreshScreen(Ally,Enemy);
                cout<<"His attack Hit! Damge Dealt: "<<Enemy.att3<<"\n\n";
                Sleep(3000);
        }
}else{
        refreshScreen(Ally,Enemy);
        cout<<"His attack missed!\n\n";
        Sleep(3000);
}
//Decrease Invisibility duration by 1
```

```
	if(invisibility.size()>1){
		invisibility.pop();
	}
	//If Invisibility is depleated, it notifies user
	if(invisibility.size()==1){
		refreshScreen(Ally,Enemy);
		cout<<"Invisibilty has worn off!";
		Sleep(3000);
		invisibility.pop();
	}

	//===== Determine if you are still alive ============
	if(Ally.health<=0){
		cout<<"You Have Been Slained by "<<Enemy.name<<"!\n\n\n
";Sleep(3000);
		string def("DEFEATED!");
		for(int i=0;i<def.size();i++){
			cout<<def[i];
		}Sleep(3000);
		system("CLS");
		return;
	}
	//Start another round
	fightingPhase(Ally,Enemy);
}

/*================================ refreshScreen
========================================
	Purpose: This function refreshes the screen and updates different properties such
		as Health, Mana, and Items
=============================================================================================
*/
void refreshScreen(Hero&Ally,Hero&Enemy){
	system("CLS");

	if(Ally.health<0)Ally.health = 0;
	if(Enemy.health<0)Enemy.health = 0;
	if(Ally.mana<0)Ally.mana=0;

	cout<<endl;

	//========= Enemy's Bars ======================
	cout << "    -"<<Enemy.name<<"-\n";
	if(Enemy.health==800){
		cout << "        Health: "<<Enemy.health/8<<"% ";
		for(int i=0;i<Enemy.health/40;i++){
			cout<<"=";
		}
	}else if((Enemy.health<800)&&(Enemy.health>100)){
		cout << "        Health:  "<<Enemy.health/8<<"% ";
		for(int i=0;i<Enemy.health/40;i++){
			cout<<"=";
		}
	}else if((Enemy.health<100)&&(Enemy.health>=20)){
		cout << "        Health:  "<<Enemy.health/8<<"% ";
		for(int i=0;i<Enemy.health/40;i++){
			cout<<"=";
		}
```

```
}else if(Enemy.health<20){
	cout << "	Health:   "<<Enemy.health/8<<"% ";
	for(int i=0;i<Enemy.health/40;i++){
		cout<<"=";
	}
}
cout << endl;
//========= Your Bars ======================
cout << "\n\n    -"<<Ally.name<<"-\n";

if(Ally.name!="Aragorn the Swordsman"){
	if(Ally.health==200){
		cout << "	Health: "<<Ally.health/2<<"% ";
		for(int i=0;i<Ally.health/10;i++){
			cout<<"=";
		}
	}else if((Ally.health<200)&&(Ally.health>=20)){
		cout << "	Health:  "<<Ally.health/2<<"% ";
		for(int i=0;i<Ally.health/10;i++){
			cout<<"=";
		}
	}else if(Ally.health<20){
		cout << "	Health:   "<<Ally.health/2<<"% ";
		for(int i=0;i<Ally.health/10;i++){
			cout<<"=";
		}
	}
}
if(Ally.name=="Aragorn the Swordsman"){
	if(Ally.armor==200){
		cout << "	 Armor: "<<Ally.armor/2<<"% ";
		for(int i=0;i<Ally.armor/10;i++){
			cout<<"=";
		}
	}else if((Ally.armor<200)&&(Ally.armor>=20)){
		cout << "	 Armor:  "<<Ally.armor/2<<"% ";
		for(int i=0;i<Ally.armor/10;i++){
			cout<<"=";
		}
	}else if(Ally.armor<20){
		cout << "	 Armor:   "<<Ally.armor/2<<"% ";
		for(int i=0;i<Ally.armor/10;i++){
			cout<<"=";
		}
	}cout<<endl;
	if(Ally.health==200){
		cout << "	Health: "<<Ally.health/2<<"% ";
		for(int i=0;i<Ally.health/10;i++){
			cout<<"=";
		}
	}else if((Ally.health<200)&&(Ally.health>=20)){
		cout << "	Health:  "<<Ally.health/2<<"% ";
		for(int i=0;i<Ally.health/10;i++){
			cout<<"=";
		}
	}else if(Ally.health<20){
		cout << "	Health:   "<<Ally.health/2<<"% ";
		for(int i=0;i<Ally.health/10;i++){
```

```
                        cout<<"=";
                    }
                }
        }
        if(Ally.name!="Aragorn the Swordsman"){
                cout<<endl;
                if(Ally.mana==100){
                cout << "          Mana: "<<Ally.mana<<"% ";
                    for(int i=0;i<Ally.mana/5;i++){
                        cout<<"=";
                    }
                }else if((Ally.mana<100)&&(Ally.mana>=10)){
                    cout << "          Mana:  "<<Ally.mana<<"% ";
                    for(int i=0;i<Ally.mana/5;i++){
                        cout<<"=";
                    }
                }else if(0<Ally.mana<10){
                    cout << "          Mana:   "<<Ally.mana<<"% ";
                    for(int i=0;i<Ally.mana/5;i++){
                        cout<<"=";
                    }
                }
        }
        cout<<endl<<endl<<endl;
}
/*================================== dots ========================================
        Purpose: This function outputes "..." with delay
================================================================================================
*/
void dots(){
        string dots("...");
        for (int i = 0; i < dots.size(); i++){
                cout << dots[i];
                Sleep(750);
        }
}
/*================================ Assign Damage
=========================================
        Purpose: This function assigns random damages to each heros' attacks. If double
damage
                     is active, it doubles the random number given.
================================================================================================
*/
void assignDamage(Hero&Ally,Hero&Enemy){
        if(doubleDamage.size()>0){
                if(Ally.name=="Aragorn the Swordsman"){
                        Ally.att1 = (rand()%30+51)*2, Ally.att2=(rand()%20+41)*2,
Ally.att3=(rand()%20+41)*2;}
                else if(Ally.name=="Legolas the Archer"){
                        Ally.att1 = (rand()%30+21)*2, Ally.att2=(rand()%15+16)*2,
Ally.att3=(rand()%10+51)*2;}
                else {Ally.att1 = (rand()%30+21)*2, Ally.att2=(rand()%90+71)*2,
Ally.att3=(rand()%40+111)*2;}
        }else{
                if(Ally.name=="Aragorn the Swordsman"){
                        Ally.att1 = rand()%30+51, Ally.att2=rand()%20+41;
Ally.att3=rand()%20+41;}
                else if(Ally.name=="Legolas the Archer"){
```

```
                    Ally.att1 = rand()%30+21, Ally.att2=rand()%15+16;
Ally.att3=rand()%10+51;}
            else {Ally.att1 = rand()%30+21, Ally.att2=rand()%90+71;
Ally.att3=rand()%40+111;}
      }
      Enemy.att1 = rand()%20+41, Enemy.att2=rand()%30+41; Enemy.att3=rand()%20+41;
}
/*================================== Item Bag
=========================================
      Purpose: This functions prompts the user what item to use. Base on what item is
chosen,
                     it executes what the items is supposed to do and returns to
fighting phase
=============================================================================================
*/
bool itemBag(Hero&Ally,Hero&Enemy){
      int i =0;
      char answer='0';

      refreshScreen(Ally,Enemy);
      cout<<"What item do you wish to use?\n\n";

      for (map<string,int>::iterator itemName=items.begin(); itemName!=items.end();
++itemName){
            cout << "    " << itemName->first << " " << itemName->second << " Left" <<
endl;
            i++;
      }
      cout<<"    7.)Back\n\n";
      cin>>answer;

      while((answer!='1')&&(answer!='2')&&(answer!='3')&&(answer!='4')&&(answer!='5')&&(
answer!='6')&&(answer!='7')){
            refreshScreen(Ally,Enemy);
            for (map<string,int>::iterator itemName=items.begin();
itemName!=items.end(); ++itemName){
                  cout << "    " <<  itemName->first << " " << itemName->second << "
Left" << endl;
                  i++;
            }
            cout<<"    7.)Back\n\n Not a Valid Choice!\n\n";
            cin>>answer;
      }
      //Using a Health Potion
      if(answer=='1'){
            if((answer=='1')&&(items["1.)Health Potion"]<1)){
                  refreshScreen(Ally,Enemy);
                  cout<<"You don't have any Health Potions!\n\n";
                  Sleep(3000);
                  itemBag(Ally,Enemy);
            }else if((answer=='1')&&(Ally.health==200)){
                  refreshScreen(Ally,Enemy);
                  cout<<"Your Health is Already Full!\n\n";
                  Sleep(3000);
                  itemBag(Ally,Enemy);
            }else{
                  items["1.)Health Potion"]-=1;
                  itemCount.pop();
```

```
                for(int i = Ally.health;i<=200;i+=20){
                        Ally.health+=20;
                        if(Ally.health>200){
                                Ally.health=200;
                        }
                        refreshScreen(Ally,Enemy);
                        Sleep(125);
                }cout<<"Your Health Has Been Restored!\n\n";
                Sleep(3000);
                return true;
        }
}
//Using a Mana Potion
if(answer=='2'){
        if((answer=='2')&&(items["2.)Mana Potion  "]<1)){
                refreshScreen(Ally,Enemy);
                cout<<"You don't have any Mana Potions!\n\n";
                Sleep(3000);
                itemBag(Ally,Enemy);
        }else if((answer=='2')&&(Ally.mana==100)){
                refreshScreen(Ally,Enemy);
                cout<<"Your Mana is Already Full!\n\n";
                Sleep(3000);
                itemBag(Ally,Enemy);
        }else{
                items["2.)Mana Potion  "]-=1;
                itemCount.pop();
                for(int i = Ally.mana;i<=100;i+=10){
                        Ally.mana+=10;
                        if(Ally.mana>100){
                                Ally.mana=100;
                        }
                        refreshScreen(Ally,Enemy);
                        Sleep(125);
                }cout<<"Your Mana Has Been Restored!\n\n";
                Sleep(3000);
                return true;
        }
}
if(answer=='3'){
        if((answer=='3')&&(items["3.)Regeneration "]<1)){
        refreshScreen(Ally,Enemy);
        cout<<"You don't have any Health Potions!\n\n";
        Sleep(3000);
        itemBag(Ally,Enemy);
}else if((answer=='3')&&(Ally.health==100)&&(Ally.mana==100)){
        refreshScreen(Ally,Enemy);
        cout<<"Your Health/Mana Are Already Full!\n\n";
        Sleep(3000);
        itemBag(Ally,Enemy);
}else{
        items["3.)Regeneration"]-=1;
        itemCount.pop();
        for(int i = Ally.health;i<=200;i+=20){
                Ally.health+=20;
                if(Ally.health>200){
                        Ally.health=200;
                }
```

```
                        refreshScreen(Ally,Enemy);
                        Sleep(125);
                }
                for(int i = Ally.mana;i<=100;i+=10){
                        Ally.mana+=10;
                        if(Ally.mana>100){
                                Ally.mana=100;
                        }
                        refreshScreen(Ally,Enemy);
                        Sleep(125);
                }cout<<"Your Health/Mana Have Been Restored!\n\n";
                Sleep(3000);
                return true;
                }
        }
        //Using a Double Damage
        if(answer=='4'){
                if((answer=='4')&&(items["4.)Double Damage"]<1)){
                        refreshScreen(Ally,Enemy);
                        cout<<"You don't have any Double Damages!\n\n";
                        Sleep(3000);
                        itemBag(Ally,Enemy);
                }else if(doubleDamage.size()>0){
                        refreshScreen(Ally,Enemy);
                        cout<<"Double Damage is already Activated!\n\n";
                        Sleep(3000);
                        itemBag(Ally,Enemy);
                }else{
                        items["4.)Double Damage"]-=1;
                        itemCount.pop();
                        doubleDamage.push(1);
                        doubleDamage.push(1);
                        doubleDamage.push(1);
                        doubleDamage.push(1);
                        refreshScreen(Ally,Enemy);
                        cout<<"Double Damage Has Been Activated for your next 3
Attacks!\n\n";
                        Sleep(3000);
                        return true;
                }
        }
        //Using a Invisibility
        if(answer=='5'){
                if((answer=='5')&&(items["5.)Invisibility "]<1)){
                        refreshScreen(Ally,Enemy);
                        cout<<"You don't have any Invisibilities!\n\n";
                        Sleep(3000);
                        itemBag(Ally,Enemy);
                }else if(invisibility.size()>0){
                        refreshScreen(Ally,Enemy);
                        cout<<"Invisibility is already Activated!\n\n";
                        Sleep(3000);
                        itemBag(Ally,Enemy);
                }else{
                        items["5.)Invisibility "]-=1;
                        itemCount.pop();
                        invisibility.push(1);
                        invisibility.push(1);
```

```
                        invisibility.push(1);
                        invisibility.push(1);
                        refreshScreen(Ally,Enemy);
                        cout<<"Invisibility Has Been Activated for the Enemy's next 3
Attacks!\n\n";
                        Sleep(3000);
                        return true;
                }
        }
        if(answer=='6'){
                if((answer=='6')&&(items["6.)Haste        "]<1)){
                        refreshScreen(Ally,Enemy);
                        cout<<"You don't have any Haste!\n\n";
                        Sleep(3000);
                        itemBag(Ally,Enemy);
                }else if(haste.size()>0){
                        refreshScreen(Ally,Enemy);
                        cout<<"Haste is already Activated!\n\n";
                        Sleep(3000);
                        itemBag(Ally,Enemy);
                }else{
                        items["6.)Haste        "]-=1;
                        itemCount.pop();
                        haste.push(1);
                        haste.push(1);
                        haste.push(1);
                        haste.push(1);
                        refreshScreen(Ally,Enemy);
                        cout<<"Haste Has Been Activated for your next 3 Attacks!\n\n";
                        Sleep(3000);
                        return true;
                }
        }
        //Quit Item Bag
        if(answer=='7'){
                fightingPhase(Ally,Enemy);
        }
}
/*================================== Get Answer
=========================================
        Purpose: This function asks and validates what the user what they want to do.
================================================================================================
*/
void getAnswer(Hero&Ally,Hero&Enemy,char&answer){
        cout<<"What do you wish to do?\n\n";
        if(Ally.name!="Aragorn the Swordsman"){
                if(Ally.name=="Gandalf the Wizard"){

        cout<<setw(7)<<"1.)"<<Ally.att1N<<endl<<setw(7)<<"2.)"<<Ally.att2N<<" (-
"<<Ally.att2M<<"% Mana)"<<endl
                                <<setw(7)<<"3.)"<<Ally.att3N<<" (-"<<Ally.att3M<<"%
Mana)"<<endl<<"    4.)Item Bag ("<<itemCount.size()<<" Items)\n\n";
                }else{

        cout<<setw(7)<<"1.)"<<Ally.att1N<<endl<<setw(7)<<"2.)"<<Ally.att2N<<" (-
"<<Ally.att2M<<"% Mana)"<<endl
                                <<setw(7)<<"3.)"<<Ally.att3N<<endl<<"    4.)Item Bag
("<<itemCount.size()<<" Items)\n\n";
```

```
		}
	}else{
		cout<<setw(7)<<"1.)"<<Ally.att1N<<endl<<setw(7)<<"2.)"<<Ally.att2N<<endl
			<<setw(7)<<"3.)"<<Ally.att3N<<endl<<"    4.)Item Bag
("<<itemCount.size()<<" Items)\n\n";
	}
	cin>>answer;

	//If mana is lower than desired move mana cost
	while(((answer=='2')&&(Ally.mana<Ally.att2M))||((Ally.name=="Gandalf the
Wizard")&&(answer=='3')&&(Ally.mana<Ally.att3M))){
		refreshScreen(Ally,Enemy);
			cout<<"What do you wish to do?\n\n";
			if(Ally.name!="Aragorn the Swordsman"){
				if(Ally.name=="Gandalf the Wizard"){

	cout<<setw(7)<<"1.)"<<Ally.att1N<<endl<<setw(7)<<"2.)"<<Ally.att2N<<" (-
"<<Ally.att2M<<"% Mana)"<<endl
							<<setw(7)<<"3.)"<<Ally.att3N<<" (-
"<<Ally.att3M<<"% Mana)"<<endl<<"    4.)Item Bag ("<<itemCount.size()<<" Items)\n\n";
				}else{

	cout<<setw(7)<<"1.)"<<Ally.att1N<<endl<<setw(7)<<"2.)"<<Ally.att2N<<" (-
"<<Ally.att2M<<"% Mana)"<<endl
							<<setw(7)<<"3.)"<<Ally.att3N<<endl<<"    4.)Item
Bag ("<<itemCount.size()<<" Items)\n\n";
				}
			}else{

	cout<<setw(7)<<"1.)"<<Ally.att1N<<endl<<setw(7)<<"2.)"<<Ally.att2N<<endl
					<<setw(7)<<"3.)"<<Ally.att3N<<endl<<"    4.)Item Bag
("<<itemCount.size()<<" Items)\n\n";
			}
			cout<<"You do not have enough Mana..\n\n";
			cin>>answer;
		if((Ally.name=="Gandalf the
Wizard")&&(answer=='3')&&(Ally.mana<Ally.att3M)){
		refreshScreen(Ally,Enemy);
		cout<<"What do you wish to do?\n\n";
		cout<<setw(7)<<"1.)"<<Ally.att1N<<endl<<setw(7)<<"2.)"<<Ally.att2N<<" (-
"<<Ally.att2M<<"% Mana)"<<endl
		<<setw(7)<<"3.)"<<Ally.att3N<<" (-"<<Ally.att3M<<"% Mana)"<<endl<<"
4.)Item Bag ("<<itemCount.size()<<" Items)\n\n";
		cout<<"You do not have enough Mana..\n\n";
		cin>>answer;
		}
	}
	//If incorrect move is input
	while((answer!='1')&&(answer!='2')&&(answer!='3')&&(toupper(answer!='4'))){
		refreshScreen(Ally,Enemy);
		cout<<"What do you wish to do?\n\n";
		if(Ally.name!="Aragorn the Swordsman"){
			if(Ally.name=="Gandalf the Wizard"){

	cout<<setw(7)<<"1.)"<<Ally.att1N<<endl<<setw(7)<<"2.)"<<Ally.att2N<<" (-
"<<Ally.att2M<<"% Mana)"<<endl
				<<setw(7)<<"3.)"<<Ally.att3N<<" (-"<<Ally.att3M<<"%
Mana)"<<endl<<"    4.)Item Bag ("<<itemCount.size()<<" Items)\n\n";
```

```
                        }else{

        cout<<setw(7)<<"1.)"<<Ally.att1N<<endl<<setw(7)<<"2.)"<<Ally.att2N<<endl
                                        <<setw(7)<<"3.)"<<Ally.att3N<<endl<<"    4.)Item Bag
("<<Ally.items<<" Items)\n\n";
                        }
                }else{

        cout<<setw(7)<<"1.)"<<Ally.att1N<<endl<<setw(7)<<"2.)"<<Ally.att2N<<endl
                                <<setw(7)<<"3.)"<<Ally.att3N<<endl<<"    4.)Item Bag
("<<itemCount.size()<<" Items)\n\n";
                }
                cout<<"Invalid choice..\n\n";
                cin>>answer;
        }
        //If you choose item bag and you dont have any items
        while((answer=='4')&&(itemCount.empty())){
                refreshScreen(Ally,Enemy);
                cout<<"What do you wish to do?\n\n";
                if(Ally.name!="Aragorn the Swordsman"){

        cout<<setw(7)<<"1.)"<<Ally.att1N<<endl<<setw(7)<<"2.)"<<Ally.att2N<<" (-
"<<Ally.att2M<<" Mana)"<<endl
                                <<setw(7)<<"3.)"<<Ally.att3N;if(Ally.name=="Gandalf the
Wizard"){cout<<" (-"<<Ally.att3M<<" Mana)";}
                        cout<<endl<<"    4.)Item Bag ("<<itemCount.size()<<" Items)\n\n";
                }else{

        cout<<setw(7)<<"1.)"<<Ally.att1N<<endl<<setw(7)<<"2.)"<<Ally.att2N<<endl
                                <<setw(7)<<"3.)"<<Ally.att3N<<endl<<"    4.)Item Bag
("<<itemCount.size()<<" Items)\n\n";
                }
                cout<<"You do not have any items!\n\n";
                cin>>answer;
        }
}
/*================================ This function
=========================================
        Purpose: This fucntion initiates specific attacks based on what the user wanted.
                         After completing this function, if double damage and haste duration
is decrease
                         by 1 if they are active.
================================================================================================
*/
void attack(Hero&Ally,Hero&Enemy,char answer){

        //Sets 25% chance of missing
        int youHit = rand()%4;
        //Sets 25% of being too slow for Aragorn
        int slowHit = rand()%4;
        //Sets random number to be used for Legolas Multiple hits
        int count = rand()%100+1;

        refreshScreen(Ally,Enemy);
        //Display what move you chose
        switch(answer){
                case '1':cout<<"You chose to attack with "<<Ally.att1N;break;
                case '2':cout<<"You chose to attack with "<<Ally.att2N;break;
```

```
			case '3':cout<<"You chose to attack with "<<Ally.att3N;break;
		}
		dots();
		system("CLS");

		if(answer!='4'){
			if(youHit<3){
				if((Ally.name=="Aragorn the Swordsman")&&(slowHit>2)){
					refreshScreen(Ally,Enemy);
					cout<<"Your Attack Was Too Slow!";
					Sleep(3000);
				}else{
				switch(toupper(answer)){
					case '1':
						Enemy.health -= Ally.att1;
						refreshScreen(Ally,Enemy);
						cout<<"Your Attack Hit! Damage Dealt:
"<<Ally.att1;

						//Write Damage to Tree
						temp = new node;
						temp->info = Ally.att1;
						bst.insert(root, temp);

						//Write Damage to Hash Table
						hashDamage(Ally.att1);
						break;
					case '2':
						if(Ally.name=="Legolas the Archer"){
							Ally.mana -= Ally.att2M;
							int tDmg = 0;
							stack<int> times;
							stack<int> totalDmg;
							if(count>0){
								times.push(1);
								totalDmg.push(Ally.att2);
								tDmg += totalDmg.top();
								Enemy.health -= Ally.att2;
								refreshScreen(Ally,Enemy);
								cout<<"Your Attack Hit
"<<times.size()<<" Times! Damage Dealt: "<<Ally.att2<<" Total Damage: "<<tDmg;
								Sleep(1500);
							}if(30<count<=60){
								assignDamage(Ally,Enemy);
								times.push(1);
								totalDmg.push(Ally.att2);
								tDmg += totalDmg.top();
								Enemy.health -= Ally.att2;
								refreshScreen(Ally,Enemy);
								cout<<"Your Attack Hit
"<<times.size()<<" Times! Damage Dealt: "<<Ally.att2<<" Total Damage: "<<tDmg;
								Sleep(1500);
							}if(60<count<=80){
								assignDamage(Ally,Enemy);
								times.push(1);
								totalDmg.push(Ally.att2);
								tDmg += totalDmg.top();
								Enemy.health -= Ally.att2;
```

```
                                                refreshScreen(Ally,Enemy);
                                                cout<<"Your Attack Hit
"<<times.size()<<" Times! Damage Dealt: "<<Ally.att2<<" Total Damage: "<<tDmg;
                                                Sleep(1500);
                                        }if(80<count<=90){
                                                assignDamage(Ally,Enemy);
                                                times.push(1);
                                                totalDmg.push(Ally.att2);
                                                tDmg += totalDmg.top();
                                                Enemy.health -= Ally.att2;
                                                refreshScreen(Ally,Enemy);
                                                cout<<"Your Attack Hit
"<<times.size()<<" Times! Damage Dealt: "<<Ally.att2<<" Total Damage: "<<tDmg;
                                                Sleep(1500);
                                        }if(count>90){
                                                assignDamage(Ally,Enemy);
                                                times.push(1);
                                                totalDmg.push(Ally.att2);
                                                tDmg += totalDmg.top();
                                                Enemy.health -= Ally.att2;
                                                refreshScreen(Ally,Enemy);
                                                cout<<"Your Attack Hit
"<<times.size()<<" Times! Damage Dealt: "<<Ally.att2<<" Total Damage: "<<tDmg;
                                                Sleep(1500);
                                        }

                                                //Write Damage to Tree
                                                temp = new node;
                                                temp->info = tDmg;
                                                bst.insert(root, temp);

                                                //Write Damage to Hash Table
                                                hashDamage(tDmg);
                                        }else{
                                                Enemy.health -= Ally.att2;Ally.mana-
=Ally.att2M;

                                                refreshScreen(Ally,Enemy);
                                                cout<<"Your Attack Hit! Damage Dealt:
"<<Ally.att2;


                                                //Write Damage to Tree
                                                temp = new node;
                                                temp->info = Ally.att2;
                                                bst.insert(root, temp);

                                                //Write Damage to Hash Table
                                                hashDamage(Ally.att2);
                                        }
                                        break;
                                case '3':
                                                Enemy.health -= Ally.att3;
                                                if(Ally.name=="Gandalf the Wizard"){
                                                        Ally.mana-=Ally.att3M;
                                                }
                                                refreshScreen(Ally,Enemy);

                                                cout<<"Your Attack Hit! Damage Dealt:
"<<Ally.att3;
```

```
                                                //Write Damage to Tree
                                                temp = new node;
                                                temp->info = Ally.att3;
                                                bst.insert(root, temp);

                                                //Write Damage to Hash Table
                                                hashDamage(Ally.att3);
                                                break;
                                        }
                                        if((Ally.name=="Legolas the
Archer")&&(answer=='2')){
                                                Sleep(1500);
                                        }else{
                                                Sleep(3000);
                                        }
                                }
                        }else{
                        switch(toupper(answer)){
                                case '2':
                                        Ally.mana-=Ally.att2M;
                                        break;
                                case '3':
                                        if(Ally.name=="Gandalf the Wizard"){
                                        Ally.mana-=Ally.att3M;
                                }
                        }
                refreshScreen(Ally,Enemy);
                if((Ally.name=="Aragorn the Swordsman")&&(slowHit>2)){
                        refreshScreen(Ally,Enemy);
                        cout<<"Your Attack Was Too Slow!";
                        Sleep(3000);
                }else{
                        cout<<"Your Attack Missed!";
                Sleep(3000);
                }
        }
        }
        //If active, Reduces double damage duration by 1
        if(doubleDamage.size()>1){doubleDamage.pop();}
        //Notifies users that DD wore off
        if(doubleDamage.size()==1){
                refreshScreen(Ally,Enemy);
                cout<<"Double Damage has worn off!";
                Sleep(3000);
                doubleDamage.pop();
        }


        //If active, Reduces haste duration by 1
        if(haste.size()>1){haste.pop();}

        //Re-attacks if haste is active
        while(haste.size()>1){
                if(Ally.mana==0){
                        answer = '1';
                }
                attack(Ally,Enemy,answer);
```

```
        }
        //Notifies users that haste wore off
        if(haste.size()==1){
                refreshScreen(Ally,Enemy);
                cout<<"Haste has worn off!";
                Sleep(3000);
                haste.pop();
        }
}
```