**Jacob Trubey**

**CSC-17A**

**Section 43950**

**Spring 2015**

**Project 2**

**Mercenary Mission**

**Introduction: Mercenary Mission**

Mercenary Mission is a turn-based/text-based strategy game involving money and mercenary warfare. You start with the options of viewing a tutorial on the game, starting the game, or viewing the list of high scores.

**Viewing The Tutorial**

Viewing the tutorial is much like viewing this write-up, it breaks down the details of the game by section and tells the user how to play.

**Starting The Game**

*Choosing an amount of mercenaries to train*

When starting the game, you are given $200,000 which you start using by picking an amount of mercenaries to train with. This is the only time during the game that you will have this option, so you will have to apply some thought or strategy here.

*Increasing Abilities/Resuscitating Mercenaries*

Whatever funds you have left over from training your team of mercenaries can then be used to increase each of your mercenaries’ abilities. These abilities include ‘attack’, which is a determining factor in dealing a lethal attack to an enemy (based on the idea of good equipment and the ability to use iteffectively), and ‘defense’, which is a determining factor on whether or not an enemy can deal a lethal attack to your mercenary (also based on the idea of equipment and personal skill). Increasing these statistics costs money, but each have their own perks in a battle sequence, and it is up to you to determine which is more worth your money for given circumstances. Another option in the ability increasing function is the option to resuscitate any injured mercenaries, but this option is naturally only useable if you have injured mercenaries.

*Choosing a job*

There are 1 of 3 types of jobs to choose from, in descending order of difficulty from lowest to highest. Each job has a risk rating that demonstrates this difficulty (low, moderate, high). Beginning players will be most likely unable to survive the moderate difficulty level, let alone the high level, so it would be a good idea to spend some time taking low difficulty level jobs of escorting dignitaries and using your monetary rewards to increase your mercenaries abilities until you feel that they are ready for the more difficult mission.

*Engaging A Mission/Battle Sequence*

The battle sequence starts when a mission is accepted and engaged. This sequence involves a random generation of enemies (The possible amounts depending on the difficultly of the mission) and then more random generation determines which of your mercenaries are personally engaged by an enemy, and even further random generation uses variations on your mercenaries base attack score, vs. the enemy’s randomly generated defense, and the enemy’s randomly generated attack score, based on your mercenaries base defense. A successful hit is all that is needed to defeat either the enemy or your mercenary (keeping in mind unsuccessful attacks are evaded/blocked by body armor, or considered negligible, and that successful attacks or those inflicted by firearms or other equally lethal weapons with enough damage to defeat the opponent). After the two attack each other simultaneously, it is then determined how much of an attack score was given by both opponents and who, if any, was defeated as a result. If all of your mercenaries are defeated before defeating all of the enemies, you fail the mission with no reward. If all of the enemies are defeated before all of your mercenaries are, then the mission is a success and your reward is earned. If you defeat the last enemy while that enemy simultaneously defeats your last mercenary, you successfully complete the mission and receive the reward, but are then unable to accept any other jobs until you resuscitate one of your mercenaries. Should you have insufficient funds to resuscitate a mercenary and all of your mercenaries are injured, it is up to you to quit the game from the “Job List” menu as your only other option is to continue to go through the other menu options to no avail.

*Saving Your Score*

After quitting the game you have the option to save your score. If you choose yes, you are then prompted to enter your initials which will precede the high score you earned from playing the game. Your high score is based on money earned, services purchased, and mission ready (non-injured) mercenaries upon quitting. This score will then be saved to a file named “score”, which will keep a log of all previously saved scores on your computer.

**Viewing High Scores**

Selecting the option to “View High Scores” allows you to view the file called “scores” which contains the initials and accompanying high scores of previously played games.

**Variables Used**

|  |  |  |
| --- | --- | --- |
| **Type** | **Variable Name** | **Purpose** |
| int | inN | Initial menu choice |
| int | funds | Money earned in game |
| int | numMrc | Initial number of mercenaries chosen |
| class Mercenary | mercenaries | Array of classes, each for individual mercenaries and their statistics (name, attack, defense, stable) |
| struct Statistics | stats | Structure variable for mercenary statistics |
| struct Army | platoon | Structure variable for amount of mercenary types |
| class Grenadier | grenadiers | Array of classes, each for individual grenadiers and their statistics |
| class Medic | medics | Array of classes, each for individual medics and their statistics |
| bool | misRed | Determines if any of your mercenaries are mission ready |
| const int | SIZE | Size of character array for player’s initials |
| char | Initials | Saves player’s initials for high score |
| int | highScore | Saves player’s high score |
| char | incChc | Choice to increase abilities |
| int | mrcChc | Choose which mercenary to improve |
| char | conChc | Confirm previous choice |
| char | ablChc | Choose which ability to improve |
| int | jobChc | Choose which job to accept |
| char | qitChc | Choice to quit the game |
| char | savChc | Choice to save high score |
| string | name | Name of mercenary in structure |
| int | attack | attack of mercenary in structure |
| int | defense | defense of mercenary in structure |
| string | type | Mercenary type |
| bool | stable | Health stability of mercenary in structure |
| unsigned | seed | Random number generator |
| int | numEnm | Number of enemies encountered |
| int | mrcEng | Mercenary engaged by enemy |
| int | mrcHit | Mercenary’s generated Hit |
| int | enmHit | Enemy’s generated Hit |
| int | enmAtk | Enemy’s base attack score |
| int | enmDfn | Enemy’s base defense score |

**Concepts from text**

Abstract Class: mercenary.h

Structure: mercenary.h, line 15

Virtual Function: mercenary.h, line 57

Inheritance: grenadier.h, line 8

Polymorphism: grenadier.h, line 46

**Project Code: Main Source File “main.cpp”**

/\*

\* File: main.cpp

\* Author: Jacob Trubey

\*

\* Created on May 27, 2015, 5:18 PM

\*/

//Library includes Here!!!

#include "grenadier.h"

#include "Medic.h"

#include <iostream>

#include <iomanip>

#include <string>

#include <cstdlib>

#include <ctime>

#include <fstream>

#include <typeinfo>

using namespace std;

//Global Constants Here!!!

//Function Prototypes Here!!!

void Menu(); //function to display main game menu

int getN(); //function to take in user's menu choice

void def(int); //function to display program exit

void viewTutorial(); //function to view game tutorial

void startGame(); //function to start game

void viewScores(); //function to view high scores

//Begin Execution Here!!!

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

int inN; //user's menu choice

do{

Menu(); //call function to display main game menu

inN=getN(); //call function to take in user's menu choice

switch(inN){ //choose from menu options

case 1: viewTutorial();break; //call function to view game tutorial

case 2: startGame();break; //call function to start game

case 3: viewScores();break; //call function to view high scores

default: def(inN);} //call function to display program exit

}while(inN>=1&&inN<=3);

return 0; //exit program

}

void Menu(){

cout<<"Welcome to Mercenary Mission!"<<endl; //display main game menu

cout<<"(a turn based strategy game of)"<<endl;

cout<<"(money and hired mercenaries)"<<endl<<endl;

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl<<endl;

cout<<"Please type the number associated with the option from the following menu list."<<endl<<endl;

cout<<"1) View Tutorial"<<endl;

cout<<"2) Start Game"<<endl;

cout<<"3) View High Scores"<<endl;

cout<<"(type anything else to exit)"<<endl<<endl;

}

int getN(){

int inN; //take in user's menu choice

cin>>inN;

cout<<endl;

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl<<endl;

return inN;

}

void viewTutorial(){

//display game tutorial

cout<<"Starting The Game: Choosing an amount of mercenaries to train: When starting the game,"<<endl

<<"you are given $200,000 which you start using by picking an amount of mercenaries to train with."<<endl

<<"This is the only time during the game that you will have this option, so you will have to apply"<<endl

<<"some thought or strategy here."<<endl<<endl

<<"Increasing Abilities/Resuscitating Mercenaries: Whatever funds you have left over from training"<<endl

<<"your team of mercenaries can then be used to increase each of your mercenaries’ abilities."<<endl

<<"These abilities include ‘attack’, which is a determining factor in dealing a lethal attack to an enemy"<<endl

<<"(based on the idea of good equipment and the ability to use it effectively), and ‘defense’, which is a"<<endl

<<"determining factor on whether or not an enemy can deal a lethal attack to your mercenary (also based"<<endl

<<"on the idea of equipment and personal skill). Increasing these statistics costs money, but each have"<<endl

<<"their own perks in a battle sequence, and it is up to you to determine which is more worth your money"<<endl

<<"for given circumstances. Another option in the ability increasing function is the option to resuscitate"<<endl

<<"any injured mercenaries, but this option is naturally only useable if you have injured mercenaries."<<endl<<endl

<<"Choosing a job: There are 1 of 3 types of jobs to choose from, in descending order of difficulty"<<endl

<<"from lowest to highest. Each job has a risk rating that demonstrates this difficulty (low, moderate, high)."<<endl

<<"Beginning players will be most likely unable to survive the moderate difficulty level, let alone the"<<endl

<<"high level, so it would be a good idea to spend some time taking low difficulty level jobs of escorting"<<endl

<<"dignitaries and using your monetary rewards to increase your mercenaries abilities until you feel that"<<endl

<<"they are ready for the more difficult mission."<<endl<<endl

<<"Engaging A Mission/Battle Sequence: The battle sequence starts when a mission is accepted and engaged."<<endl

<<"This sequence involves a random generation of enemies (The possible amounts depending on the difficultly"<<endl

<<"of the mission) and then more random generation determines which of your mercenaries are personally"<<endl

<<"engaged by an enemy, and even further random generation uses variations on your mercenaries base attack"<<endl

<<"score, vs. the enemy’s randomly generated defense, and the enemy’s randomly generated attack score, based"<<endl

<<"on your mercenaries base defense. A successful hit is all that is needed to defeat either the enemy or"<<endl

<<"your mercenary (keeping in mind unsuccessful attacks are evaded/blocked by body armor, or considered"<<endl

<<"negligible, and that successful attacks or those inflicted by firearms or other equally lethal weapons"<<endl

<<"with enough damage to defeat the opponent). After the two attack each other simultaneously, it is then"<<endl

<<"determined how much of an attack score was given by both opponents and who, if any, was defeated as a"<<endl

<<"result. If all of your mercenaries are defeated before defeating all of the enemies, you fail the mission"<<endl

<<"with no reward. If all of the enemies are defeated before all of your mercenaries are, then the mission"<<endl

<<"is a success and your reward is earned. If you defeat the last enemy while that enemy simultaneously defeats"<<endl

<<"your last mercenary, you successfully complete the mission and receive the reward, but are then unable to"<<endl

<<"accept any other jobs until you resuscitate one of your mercenaries. Should you have insufficient funds"<<endl

<<"to resuscitate a mercenary and all of your mercenaries are injured, it is up to you to quit the game from"<<endl

<<"the “Job List” menu as your only other option is to continue to go through the other menu options to no avail."<<endl<<endl

<<"Saving Your Score: After quitting the game you have the option to save your score. If you choose yes, you are"<<endl

<<"then prompted to enter your initials which will precede the high score you earned from playing the game."<<endl

<<"Your high score is based on money earned, services purchased, and mission ready (non-injured) mercenaries"<<endl

<<"upon quitting. This score will then be saved to a file named “score”, which will keep a log of all previously"<<endl

<<"saved scores on your computer."<<endl<<endl

<<"Viewing High Scores: Selecting the option to “View High Scores” allows you to view the file called “scores”"<<endl

<<"which contains the initials and accompanying high scores of previously played games."<<endl<<endl;

}void startGame(){

int funds = 200000; //starting funds

int numMrc; //size of mercenary array

bool misRed = true; //boolean flag to determine if mercenary group is ready for a mission

//Medic medics[];

//Grenadier grenadiers[];

Mercenary::platoon = Mercenary::amtMrc(&funds, &numMrc); //call function to calculate amount of mercenaries

int numGren = Mercenary::platoon.numGren; //number of grenadiers

int numMedic = Mercenary::platoon.numMedic; //number of medics

int total = numGren + numMedic; //total amount of mercenaries

//Mercenary mercenaries[total];

//Mercenary\* mercenaries = new Mercenary[total];

Mercenary\* mercenaries; //dynamically allocate mercenary array

Grenadier grenadiers[numGren]; //array of grenadiers

Medic medics[numMedic]; //array of medics

numMrc = total; //total amount of mercenaries

mercenaries = new Grenadier[numGren];

// int i = 0;

//

// for(i = 0; i < numGren; i++){ //fill mercenary array with grenadiers

// mercenaries[i] = &grenadiers[i];

// }

//

// for(i = 0; i < numMedic; i++){ //fill mercenary array with medics

// mercenaries[i + numGren] = &medics[i];

// }

(mercenaries)->trainMrc(&funds, &numMrc, mercenaries); //call function to set base stats of mercenaries

(mercenaries)->incAbl(&funds, &numMrc, mercenaries); //call function to increase stats of mercenaries

(mercenaries)->chooseJob(&funds, &numMrc, &misRed, mercenaries); //call function to choose job for mercenary group

(mercenaries)->saveScore(&funds,&numMrc, mercenaries); //call function to save score

cout<<"Thank you for playing."<<endl<<endl;

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl<<endl;

delete [] mercenaries; //deallocate memory

mercenaries = 0;

}

void viewScores(){

const int SIZE = 3; //size of initials array

char initials[SIZE] = {' ',' ',' '}; //array of initials for saved scores

int highScore = 0; //high score of saved games

fstream file;

file.open("scores.txt", ios::in | ios::binary); //open file of saved scores

cout<<"High Scores"<<endl<<endl;

file.read(initials, sizeof(initials)); //read in first set of initials

file.read(reinterpret\_cast<char \*>(&highScore), sizeof(highScore)); //read in first high score

while (!file.eof()){ //display first set of initials and high score and read in subsequent sets of initials and high scores

for(int count = 0; count < SIZE; count++)

cout<<initials[count];

cout<<" "<<highScore<<endl;

file.read(initials, sizeof(initials));

file.read(reinterpret\_cast<char \*>(&highScore), sizeof(highScore));

}

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl<<endl;

file.close(); //close file of saved scores

}

void def(int inN){

cout<<"Exiting the program..."<<endl; //display program exit

}

**Project Code: Mercenary Class Header File “mercenary.h”**

/\*

\* File: mercenary.h

\* Author: Jacob Trubey

\*

\* Created on May 4, 2015, 7:11 PM

\*/

#include <iostream>

#include <cstdlib>

#include <string>

using namespace std;

#ifndef MERCENARY\_H

#define MERCENARY\_H

struct Mercenary{

string name;

int attack;

int defense;

bool stable;

};

#endif /\* MERCENARY\_H \*/

**Project Code: Mercenary Class CPP File “mercenary.cpp”**

#include "Mercenary.h"

#include <ctime>

#include <cstdlib>

#include <fstream>

#include <typeinfo>

Mercenary::Army Mercenary::platoon;

Mercenary::Mercenary(){ //initialize values for Mercenary class

stats.attack = 0;

stats.defense = 0;

stats.name = "merc";

stats.type = "Mercenary";

stats.stable = true;

}

Mercenary::Mercenary(std::string n){

stats.attack = 0;

stats.defense = 0;

stats.name = n;

stats.type = "Mercenary";

stats.stable = true;

abilityUsed = false;

}

Mercenary::Mercenary(int a, int b){

stats.attack = a;

stats.defense = b;

stats.name = "merc";

stats.type = "Mercenary";

stats.stable = true;

}

Mercenary::Mercenary(int a, int b, std::string n){

stats.attack = a;

stats.defense = b;

stats.name = n;

stats.type = "Mercenary";

stats.stable = true;

}

Mercenary::Mercenary(int a, int b, std::string n, bool s){

stats.attack = a;

stats.defense = b;

stats.name = n;

stats.type = "Mercenary";

stats.stable = s;

}

Mercenary::~Mercenary(){} //Destructor

void Mercenary::setAttack(int a){ //input validation for negative numbers for attack entered user

if(a < 0){

a = 0;

}

stats.attack = a;

}

void Mercenary::setDefense(int d){ //input validation for negative numbers for defense entered user

if(d < 0){

d = 0;

}

stats.defense = d;

}

std::string Mercenary::getType(){ //access mercenary type

return stats.type;

}

int Mercenary::getAttack(){ //access mercenary attack

return stats.attack;

}

int Mercenary::getDefense(){ //access mercenary defense

return stats.defense;

}

void Mercenary::display(){ //display mercenary info

cout << "Name: " << stats.name << endl

<< "Attack: " << stats.attack << endl

<< "Defense: " << stats.defense << endl

<< "Stable?: " << (stats.stable ? "true" : "false");

}

bool Mercenary::useAbility(int&, Mercenary\* mercs){ //virtual function for derived classes special abilities

cout<< "Ability being used.." <<endl<<endl;

//cout << endl << getType() << endl;

}

std::string Mercenary::getName(){ //access mercenary name

return stats.name;

}

Mercenary::Army Mercenary::amtMrc(int \*funds, int \*numMrc){

cout<<"Your current funds are $"<<\*funds<<"."<<endl; //display current funds and prompt user to create mercenaries

cout<<"(each mercenary costs $20000 to train)"<<endl;

cout<<"How many grenadiers do you want to train?"<<endl;

std::cin>>\*numMrc; //take in amount of grenadiers from user

cout<<endl;

if (\*numMrc<0 || \*numMrc>10){ //validate input amount of grenadiers

do{

cout<<"That is an invalid amount."<<endl;

cout<<"Please enter a valid amount of grenadiers to train."<<endl;

std::cin>>\*numMrc;

cout<<endl;

}while(\*numMrc<1 || \*numMrc>10);

}

Mercenary::platoon.numGren = \*numMrc; //set amount of grenadiers

\*funds = \*funds - 20000\*(\*numMrc); //decrement funds

cout<<"Your current funds are $"<<\*funds<<"."<<endl<<endl; //display current funds

// cout<<"How many medics do you want to train?"<<endl; //take in amount of medics from user

// std::cin>>\*numMrc;

// cout<<endl;

// if (\*numMrc<0 || \*numMrc>10){ //validate input amount of medics

// do{

// cout<<"That is an invalid amount."<<endl;

// cout<<"Please enter a valid amount of medics to train."<<endl;

// std::cin>>\*numMrc;

// cout<<endl;

// }while(\*numMrc<1 || \*numMrc>10);

// }

//

// Mercenary::platoon.numMedic = \*numMrc; //set amount of medics

Mercenary::platoon.numMedic = 0;

//\*funds = \*funds - 20000\*(\*numMrc); //decrement funds

cout<<"Your current funds are $"<<\*funds<<"."<<endl<<endl; //display current funds

return platoon; //return mercenary info

}

void Mercenary::trainMrc(int \*funds, int \*numMrc, Mercenary \*mercenaries){

std::cin.ignore();

cout << "\nNumber mercenaries: " << \*numMrc << endl;

for(int count = 0; count < \*numMrc; count++){ //set stats for group of mercenaries

cout<<"\nPlease enter mercenary number "<<count+1<<"'s name."<<endl;

getline(std::cin, (\*(mercenaries + count)).stats.name);

cout << endl;

(\*(mercenaries + count)).stats.defense = 5;

(\*(mercenaries + count)).stats.attack = 3;

(\*(mercenaries + count)).stats.stable = true;

}

cout<<"Training "<<\*numMrc<<" mercenaries..."<<endl<<endl;

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl<<endl;

cout<<"Your team of mercenaries is as follows:"<<endl<<endl; //display stats for group of mercenaries

for(int count = 0; count < \*numMrc; count++){

cout<<(\*(mercenaries + count)).stats.name<<endl;

cout<<"stats.defense: "<<(\*(mercenaries + count)).stats.defense<<endl;

cout<<"stats.attack: "<<(\*(mercenaries + count)).stats.attack<<endl;

if ((\*(mercenaries + count)).stats.stable == true)

cout<<"health status: mission ready"<<endl;

else

cout<<"health status: injured"<<endl<<endl;

cout<<"type: "<<(\*(mercenaries + count)).stats.type<<endl<<endl;

}

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl<<endl;

}

void Mercenary::incAbl(int \*funds, int \*numMrc, Mercenary \*mercenaries){

char incChc; //choice to increase abilities of a mercenary

int mrcChc; //choice of mercenary to increase abilities of

char conChc; //confirm choice of which mercenary's abilities to increase

char ablChc; //choice of ability to increase

cout<<"Would you like to improve the equipment of any of your mercenaries or resuscitate them?"<<endl;

cout<<"(improving defensive equipment increases a mercenary's defense and costs $20000 per mercenary)"<<endl;

cout<<"{improving offensive equipment increases a mercenary's attack and costs $50000 per mercenary)"<<endl;

cout<<"(resuscitation changes a mercenary's status from injured to mission ready and costs $100000 per mercenary)"<<endl;

cout<<"Please enter Y for yes, or anything else for no."<<endl;

std::cin>>incChc;

cout<<endl;

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl<<endl;

while(incChc == 'Y'){

cout<<"Which mercenary would you like to equip or resuscitate?"<<endl;

cout<<"Enter number from list associated with mercenary."<<endl<<endl;

for(int count = 0; count < \*numMrc; count++){

cout<<count + 1<<") "<<(\*(mercenaries + count)).stats.name<<endl;

cout<<"defense: "<<(\*(mercenaries + count)).stats.defense<<endl;

cout<<"attack: "<<(\*(mercenaries + count)).stats.attack<<endl;

if ((\*(mercenaries + count)).stats.stable == true)

cout<<"health status: mission ready"<<endl<<endl;

else

cout<<"health status: injured"<<endl<<endl;

cout<<"type: "<<(\*(mercenaries + count)).stats.type<<endl;

}

std::cin>>mrcChc;

cout<<endl;

if (mrcChc<1 || mrcChc>(\*numMrc)){

do{

cout<<"That is an invalid entry."<<endl;

cout<<"Please enter a valid entry from the list of mercenaries."<<endl;

std::cin>>mrcChc;

cout<<endl;

}while(mrcChc<1 || mrcChc>(\*numMrc));

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl<<endl;

}

cout<<"You've chosen to increase the abilities of "<<(\*(mercenaries + mrcChc - 1)).stats.name<<". Is this correct?"<<endl;

cout<<"Please enter Y for yes, or anything else for no."<<endl;

std::cin>>conChc;

cout<<endl;

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl<<endl;

while(conChc == 'Y'){

cout<<"What would you like to improve for "<<(\*(mercenaries + mrcChc - 1)).stats.name<<"?"<<endl;

cout<<"(please enter the letter associated with the option from the list, or anything else to go back)"<<endl<<endl;

cout<<"Your current funds are $"<<\*funds<<"."<<endl<<endl;

cout<<"D) Improve Defensive Equipment: $20000"<<endl;

cout<<"O) Improve Offensive Equipment: $50000"<<endl;

cout<<"R) Resuscitate: $100000"<<endl;

std::cin>>ablChc;

cout<<endl;

if (ablChc == 'D'){

if (\*funds < 20000){

cout<<"You have insufficient funds."<<endl<<endl;

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl<<endl;

}

else{

(\*(mercenaries + mrcChc - 1)).stats.defense++;

\*funds = \*funds - 20000;

cout<<(\*(mercenaries + mrcChc - 1)).stats.name<<"'s defense is now "<<(\*(mercenaries + mrcChc - 1)).stats.defense<<"."<<endl;

cout<<"Your current funds are $"<<\*funds<<"."<<endl<<endl;

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl<<endl;

}

}

else if (ablChc == 'O'){

if (\*funds < 50000){

cout<<"You have insufficient funds."<<endl<<endl;

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl<<endl;

}

else{

(\*(mercenaries + mrcChc - 1)).stats.attack++;

\*funds = \*funds - 50000;

cout<<(\*(mercenaries + mrcChc - 1)).stats.name<<"'s attack is now "<<(\*(mercenaries + mrcChc - 1)).stats.attack<<"."<<endl;

cout<<"Your current funds are $"<<\*funds<<"."<<endl<<endl;

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl<<endl;

}

}

else if (ablChc == 'R'){

if ((\*(mercenaries + mrcChc - 1)).stats.stable == true){

cout<<(\*(mercenaries + mrcChc - 1)).stats.name<<" is already mission ready."<<endl<<endl;

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl<<endl;

}

else if (\*funds < 100000){

cout<<"You have insufficient funds."<<endl<<endl;

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl<<endl;

}

else{

(\*(mercenaries + mrcChc - 1)).stats.stable = true;

\*funds = \*funds - 100000;

cout<<(\*(mercenaries + mrcChc - 1)).stats.name<<" has been resuscitated back to stable health."<<endl;

cout<<"Your current funds are $"<<\*funds<<"."<<endl<<endl;

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl<<endl;

}

}

conChc = 'N';

}

cout<<"Would you like to improve the equipment of any of your mercenaries or resuscitate them?"<<endl;

cout<<"{improving defensive equipment increases a mercenary's stats.defense and costs $20000 per mercenary)"<<endl;

cout<<"(improving offensive equipment increases a mercenary's stats.attack and costs $50000 per mercenary)"<<endl;

cout<<"(resuscitation changes a mercenary's status from injured to mission ready and costs $100000 per mercenary)"<<endl;

cout<<"Please enter Y for yes, or anything else for no."<<endl;

std::cin>>incChc;

cout<<endl;

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl<<endl;

}

}

void Mercenary::chooseJob(int \*funds, int \*numMrc, bool \*misRed, Mercenary \*mercenaries){

int jobChc; //job to be chosen for mercenary team

char conChc; //confirm choice of job

char qitChc = 'Y';

do{

cout<<"Please choose from the list of available mercenary missions."<<endl;

cout<<"(enter the number associated with the option from the list)"<<endl;

cout<<"(entering any number not on the list will exit the job menu)"<<endl<<endl;

cout<<"1) Escort Dignitary."<<endl;

cout<<"Risk Factor: low"<<endl;

cout<<"Payment: $20000."<<endl<<endl;

cout<<"2) Apprehend Crime Lord."<<endl;

cout<<"Risk Factor: medium"<<endl;

cout<<"Payment: $100000."<<endl<<endl;

cout<<"3) Infiltrate Foreign Military Complex."<<endl;

cout<<"Risk Factor: high"<<endl;

cout<<"Payment: $1000000."<<endl<<endl;

std::cin>>jobChc;

cout<<endl;

while(jobChc > 0 && jobChc < 4){

if(jobChc == 1){

cout<<"Escort Dignitary: $20000"<<endl<<endl;

cout<<"A wealthy dignitary is looking for security for hire."<<endl;

cout<<"As he is valuable enough to be concerned with his safety,"<<endl;

cout<<"but not in a governmental position to qualify for"<<endl;

cout<<"formal military protection, he has turned to our"<<endl;

cout<<"employers for assistance on the matter. Your job is"<<endl;

cout<<"to escort the client to his destination unharmed. Our"<<endl;

cout<<"intel has determined the risk factor of this mission to"<<endl;

cout<<"be of a low rating, though some conflict may be likely."<<endl<<endl;

cout<<"Would you like to take this job?"<<endl;

cout<<"(please enter Y for yes, or anything else to decline)"<<endl;

std::cin>>conChc;

cout<<endl;

if(conChc == 'Y'){

\*misRed = false; //mission ready boolean

for(int count = 0; count < \*numMrc; count++){

if ((\*(mercenaries + count)).stats.stable == true)

\*misRed = true;

}

if(\*misRed == false)

cout<<"You have no mission ready mercenaries. You cannot accept any jobs at this time."<<endl<<endl;

else{

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl<<endl;

mercenaries->startJob1(funds, numMrc, misRed, mercenaries);

}

}

mercenaries->incAbl(funds, numMrc, mercenaries);

}

else if(jobChc == 2){

cout<<"Apprehend Crime Lord: $100000"<<endl<<endl;

cout<<"An organized crime lord has been under surveillance of"<<endl;

cout<<"federal law enforcement for some time, but due to legal"<<endl;

cout<<"constraints, they have not been able to get close enough"<<endl;

cout<<"to him to expose his activity and try him for his crimes."<<endl;

cout<<"This is where you come in. Your mission is to enter his"<<endl;

cout<<"compound where you will find the necessary information to"<<endl;

cout<<"expose and apprehend him. Our intel has determined the"<<endl;

cout<<"risk factor for this mission to be of a moderate rating."<<endl;

cout<<"As the compound is well guarded, conflict is more than likely."<<endl<<endl;

cout<<"Would you like to take this job?"<<endl;

cout<<"(please choose Y for yes, or anything else to decline)"<<endl;

std::cin>>conChc;

cout<<endl;

if(conChc == 'Y'){

\*misRed = false; //mission ready boolean

for(int count = 0; count < \*numMrc; count++){

if ((\*(mercenaries + count)).stats.stable == true)

\*misRed = true;

}

if(\*misRed == false)

cout<<"You have no mission ready mercenaries. You cannot accept any jobs at this time."<<endl<<endl;

else{

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl<<endl;

mercenaries->startJob2(funds, numMrc, misRed, mercenaries);

}

}

mercenaries->incAbl(funds, numMrc, mercenaries);

}

else if(jobChc == 3){

cout<<"Infiltrate Foreign Military Complex: $1000000"<<endl<<endl;

cout<<"Federal Intel has determined that a foreign military power"<<endl;

cout<<"threatens the safety of the nation, but foreign policy"<<endl;

cout<<"currently prohibits government forces from intervening."<<endl;

cout<<"Our employers have been called upon to eliminate this threat."<<endl;

cout<<"Your mission is to infiltrate the foreign military complex"<<endl;

cout<<"in question and neutralize this threat. Our intel has"<<endl;

cout<<"determined that the risk factor for this mission is high, due"<<endl;

cout<<"to heavily armed military forces, making conflict almost certain."<<endl<<endl;

cout<<"Would you like to take this job?"<<endl;

cout<<"(please choose Y for yes, or anything else to decline)"<<endl;

std::cin>>conChc;

cout<<endl;

if(conChc == 'Y'){

\*misRed = false; //mission ready boolean

for(int count = 0; count < \*numMrc; count++){

if ((\*(mercenaries + count)).stats.stable == true)

\*misRed = true;

}

if(\*misRed == false)

cout<<"You have no mission ready mercenaries. You cannot accept any jobs at this time."<<endl<<endl;

else{

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl<<endl;

mercenaries->startJob3(funds, numMrc, misRed, mercenaries);

}

}

mercenaries->incAbl(funds, numMrc, mercenaries);

}

cout<<"Please choose from the list of available mercenary missions."<<endl;

cout<<"(enter the number associated with the option from the list)"<<endl;

cout<<"(entering a number not on the list will exit the job menu)"<<endl<<endl;

cout<<"1) Escort Dignitary."<<endl;

cout<<"Risk Factor: low"<<endl;

cout<<"Payment: $20000."<<endl<<endl;

cout<<"2) Apprehend Crime Lord."<<endl;

cout<<"Risk Factor: medium"<<endl;

cout<<"Payment: $100000."<<endl<<endl;

cout<<"3) Infiltrate Foreign Military Complex."<<endl;

cout<<"Risk Factor: high"<<endl;

cout<<"Payment: $1000000."<<endl<<endl;

std::cin>>jobChc;

cout<<endl;

}

cout<<"Do you wish to exit the game?"<<endl;

cout<<"(please enter Y for yes, or anything else to decline)"<<endl;

std::cin>>qitChc;

cout<<endl;

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl<<endl;

}while(qitChc != 'Y');

}

void Mercenary::startJob1(int \*funds, int \*numMrc, bool \*misRed, Mercenary \*mercenaries){

cout<<"Job accepted. Engaging mission..."<<endl<<endl;

unsigned seed = time(0);

srand(seed);

int numEnm;

int mrcEng;

int mrcHit;

int enmHit;

int enmAtk;

int enmDfn;

numEnm = 0 + rand() % 6;

cout<<"Your mercenaries encounter "<<numEnm<<" hired thugs..."<<endl<<endl;

std::cin.ignore();

cout<<"(please press enter to continue)"<<endl;

std::cin.get();

cout<<endl;

if(numEnm == 0){

cout<<"The dignitary made it to his destination unharmed. Mission accomplished!"<<endl<<endl;

\*funds = \*funds + 20000;

cout<<"Your funds are now $"<<\*funds<<"."<<endl<<endl;

}

while ((numEnm > 0) && (\*misRed == true)){

do{ //check to make sure mercenary engaged is not injured

mrcEng = (0 + rand () % (\*numMrc));

}while((\*(mercenaries + mrcEng)).stats.stable == false);

enmAtk = 1 + rand () % 3;

enmDfn = 1 + rand() % 10;

cout<<"A hired thug stats.attacks "<<(\*(mercenaries + mrcEng)).stats.name<<"!"<<endl;

mrcHit = (0 + rand() % 4)\*((\*(mercenaries + mrcEng)).stats.attack);

enmHit = (0 + rand() % 4)\*(enmAtk);

cout<<(\*(mercenaries + mrcEng)).stats.name<<" deals "<<mrcHit<<" points of damage to the thug..."<<endl;

cout<<"The thug deals "<<enmHit<<" points of damage to "<<(\*(mercenaries + mrcEng)).stats.name<<"..."<<endl;

if (mrcHit > enmDfn){

numEnm = numEnm - 1;

//(mercenaries + mrcEng)->useAbility(numEnm);

cout<<(\*(mercenaries + mrcEng)).stats.name<<" defeated the hired thug!"<<endl;

(mercenaries + mrcEng)->useAbility(numEnm, mercenaries);

}

if (enmHit > (\*(mercenaries + mrcEng)).stats.defense){

(\*(mercenaries + mrcEng)).stats.stable = false;

cout<<(\*(mercenaries + mrcEng)).stats.name<<" is now injured and won't be mission ready until resuscitated!"<<endl<<endl;

}

cout<<numEnm<<" thugs remain..."<<endl<<endl;

cout<<"(please press enter to continue)"<<endl;

std::cin.get();

cout<<endl;

\*misRed = false; //mission ready boolean

for(int count = 0; count < \*numMrc; count++){

if ((\*(mercenaries + count)).stats.stable == true)

\*misRed = true;

}

if (\*misRed == false && numEnm == 0){

cout<<"All of the hired thugs have been defeated, but all of your mercenaries are injured."<<endl;

cout<<"You have completed the mission and the dignitary made it to his destination unharmed,"<<endl;

cout<<"but are unable to accept any other jobs until one of your mercenaries are resuscitated."<<endl<<endl;

\*funds = \*funds + 20000;

cout<<"Your funds are now $"<<\*funds<<"."<<endl<<endl;

}

else if(\*misRed == false)

cout<<"All of your mercenaries are injured. You have failed the mission..."<<endl<<endl;

else if(numEnm == 0){

cout<<"All of the hired thugs have been defeated! The dignitary has made it"<<endl;

cout<<"to his destination unharmed. Mission accomplished!"<<endl<<endl;

\*funds = \*funds + 20000;

cout<<"Your funds are now $"<<\*funds<<"."<<endl<<endl;

}

}

cout<<"(please press enter to continue)"<<endl;

std::cin.get();

cout<<endl;

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl<<endl;

}

void Mercenary::startJob2(int \*funds, int \*numMrc, bool \*misRed, Mercenary \*mercenaries){

cout<<"Job accepted. Engaging mission..."<<endl<<endl;

unsigned seed = time(0);

srand(seed);

int numEnm;

int mrcEng;

int mrcHit;

int enmHit;

int enmAtk;

int enmDfn;

numEnm = 0 + rand() % 11;

cout<<"Your mercenaries encounter "<<numEnm<<" compound guards..."<<endl<<endl;

std::cin.ignore();

cout<<"(please press enter to continue)"<<endl;

std::cin.get();

cout<<endl;

if(numEnm == 0){

cout<<"The crime lord has been apprehended along with the"<<endl;

cout<<"evidence necessary for his incarceration. Mission accomplished!"<<endl<<endl;

\*funds = \*funds + 50000;

cout<<"Your funds are now $"<<\*funds<<"."<<endl<<endl;

}

while ((numEnm > 0) && (\*misRed == true)){

do{ //check to make sure mercenary engaged is not injured

mrcEng = (0 + rand () % (\*numMrc));

}while((\*(mercenaries + mrcEng)).stats.stable == false);

enmAtk = 1 + rand () % 5;

enmDfn = 1 + rand() % 20;

cout<<"A guard stats.attacks "<<(\*(mercenaries + mrcEng)).stats.name<<"!"<<endl;

mrcHit = (0 + rand() % 6)\*((\*(mercenaries + mrcEng)).stats.attack);

enmHit = (0 + rand() % 6)\*(enmAtk);

//if (medAbl == true)

cout<<(\*(mercenaries + mrcEng)).stats.name<<" deals "<<mrcHit<<" points of damage to the guard..."<<endl;

cout<<"The guard deals "<<enmHit<<" points of damage to "<<(\*(mercenaries + mrcEng)).stats.name<<"..."<<endl;

if (mrcHit > enmDfn){

numEnm = numEnm - 1;

cout<<(\*(mercenaries + mrcEng)).stats.name<<" defeated the compound gaurd!"<<endl;

}

if (enmHit > (\*(mercenaries + mrcEng)).stats.defense){

(\*(mercenaries + mrcEng)).stats.stable = false;

cout<<(\*(mercenaries + mrcEng)).stats.name<<" is now injured and won't be mission ready until resuscitated!"<<endl<<endl;

}

cout<<numEnm<<" guards remain..."<<endl<<endl;

cout<<"(please press enter to continue)"<<endl;

std::cin.get();

cout<<endl;

\*misRed = false; //mission ready boolean

for(int count = 0; count < \*numMrc; count++){

if ((\*(mercenaries + count)).stats.stable == true)

\*misRed = true;

}

if (\*misRed == false && numEnm == 0){

cout<<"All of the compound guards have been defeated, but all of your mercenaries are injured."<<endl;

cout<<"You have completed the mission and the crime lord has been apprehended along with the,"<<endl;

cout<<"evidence necessary for his incarceration, but you are unable to accept any other jobs until"<<endl;

cout<<"one of your mercenaries are resuscitated."<<endl<<endl;

\*funds = \*funds + 50000;

cout<<"Your funds are now $"<<\*funds<<"."<<endl<<endl;

}

else if(\*misRed == false)

cout<<"All of your mercenaries are injured. You have failed the mission..."<<endl<<endl;

else if(numEnm == 0){

cout<<"All of the compound guards have been defeated! The crime lord has been apprehended"<<endl;

cout<<"along with the evidence necessary for his incarceration. Mission accomplished!"<<endl<<endl;

\*funds = \*funds + 50000;

cout<<"Your funds are now $"<<\*funds<<"."<<endl<<endl;

}

}

cout<<"(please press enter to continue)"<<endl;

std::cin.get();

cout<<endl;

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl<<endl;

}

void Mercenary::startJob3(int \*funds, int \*numMrc, bool \*misRed, Mercenary \*mercenaries){

cout<<"Job accepted. Engaging mission..."<<endl<<endl;

unsigned seed = time(0);

srand(seed);

int numEnm;

int mrcEng;

int mrcHit;

int enmHit;

int enmAtk;

int enmDfn;

numEnm = 0 + rand() % 21;

cout<<"Your mercenaries encounter "<<numEnm<<" military soldiers..."<<endl<<endl;

std::cin.ignore();

cout<<"(please press enter to continue)"<<endl;

std::cin.get();

cout<<endl;

if(numEnm == 0){

cout<<"The military complex has been successfully infiltrated and"<<endl;

cout<<"its threat has been neutralized. Mission accomplished!"<<endl<<endl;

\*funds = \*funds + 100000;

cout<<"Your funds are now $"<<\*funds<<"."<<endl<<endl;

}

while ((numEnm > 0) && (\*misRed == true)){

do{ //check to make sure mercenary engaged is not injured

mrcEng = (0 + rand () % (\*numMrc));

}while((\*(mercenaries + mrcEng)).stats.stable == false);

enmAtk = 1 + rand () % 10;

enmDfn = 1 + rand() % 40;

cout<<"A soldier stats.attacks "<<(\*(mercenaries + mrcEng)).stats.name<<"!"<<endl;

mrcHit = (0 + rand() % 11)\*((\*(mercenaries + mrcEng)).stats.attack);

enmHit = (0 + rand() % 11)\*(enmAtk);

cout<<(\*(mercenaries + mrcEng)).stats.name<<" deals "<<mrcHit<<" points of damage to the soldier..."<<endl;

cout<<"The soldier deals "<<enmHit<<" points of damage to "<<mercenaries[mrcEng].stats.name<<"..."<<endl;

if (mrcHit > enmDfn){

numEnm = numEnm - 1;

cout<<(\*(mercenaries + mrcEng)).stats.name<<" defeated the military soldier!"<<endl;

}

if (enmHit > (\*(mercenaries + mrcEng)).stats.defense){

(\*(mercenaries + mrcEng)).stats.stable = false;

cout<<(\*(mercenaries + mrcEng)).stats.name<<" is now injured and won't be mission ready until resuscitated!"<<endl<<endl;

}

cout<<numEnm<<" soldiers remain..."<<endl<<endl;

cout<<"(please press enter to continue)"<<endl;

std::cin.get();

cout<<endl;

\*misRed = false; //mission ready boolean

for(int count = 0; count < \*numMrc; count++){

if ((\*(mercenaries + count)).stats.stable == true)

\*misRed = true;

}

if (\*misRed == false && numEnm == 0){

cout<<"All of the military soldiers have been defeated, but all of your mercenaries are injured."<<endl;

cout<<"You have completed the mission and the military complex has been infiltrated and its threat has been"<<endl;

cout<<"neutralized, but you are unable to accept any other jobs until one of your mercenaries are resuscitated."<<endl<<endl;

\*funds = \*funds + 100000;

cout<<"Your funds are now $"<<\*funds<<"."<<endl<<endl;

}

else if(\*misRed == false)

cout<<"All of your mercenaries are injured. You have failed the mission..."<<endl<<endl;

else if(numEnm == 0){

cout<<"All of the military soldiers have been defeated! The military complex has been"<<endl;

cout<<"successfully infiltrated and its threat has been neutralized. Mission accomplished!"<<endl<<endl;

\*funds = \*funds + 100000;

cout<<"Your funds are now $"<<\*funds<<"."<<endl<<endl;

}

}

cout<<"(please press enter to continue)"<<endl;

std::cin.get();

cout<<endl;

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl<<endl;

}

void Mercenary::saveScore(int \*funds, int \*numMrc, Mercenary \*mercenaries){

char savChc = 'Y';

const int SIZE = 3;

char initials[SIZE] = {' ',' ',' '};

int highScore = 0;

std::fstream file;

file.open("scores.txt", std::ios::out | std::ios::binary | std::ios::app);

cout<<"Would you like to save your score?"<<endl;

cout<<"(please enter Y for yes, or anything else to decline)"<<endl;

std::cin>>savChc;

cout<<endl;

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl<<endl;

if(savChc == 'Y'){

cout<<"(please enter your initials to be saved with your high score)"<<endl;

for(int count = 0; count < SIZE; count++)

std::cin>>initials[count];

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl<<endl;

cout<<endl<<endl;

for(int count = 0; count < \*numMrc; count++){

highScore = highScore + 20000\*((\*(mercenaries + count)).stats.defense - 5);

highScore = highScore + 20000\*((\*(mercenaries + count)).stats.attack - 3);

if ((\*(mercenaries + count)).stats.stable == true)

highScore = highScore + 20000;

}

highScore = highScore + \*funds;

file.write(initials, sizeof(initials));

file.write(reinterpret\_cast<char \*>(&highScore), sizeof(highScore));

cout<<endl<<endl;

}

file.close();

}

void changeStat(int option, Mercenary& merc){

if(option == 1){

merc.stats.attack = 12;

}

else if(option == 2){

merc.stats.defense = 23;

}

else if(option == 3){

merc.stats.stable = true;

}

else{}

}

Mercenary::Statistics Mercenary::getStats(){

return stats;

}

**Project Code: Grenadier Class Header File “grenadier.h”**

#ifndef Grenadier\_H\_

#define Grenadier\_H\_

#include "Mercenary.h"

#include <iostream>

#include <string>

class Grenadier : public Mercenary //inherited grenadier mercenary class

{

public:

Grenadier();

Grenadier(std::string);

Grenadier(int a, int b);

Grenadier(int a, int b, std::string n);

Grenadier(int a, int b, std::string n, bool s);

~Grenadier();

bool useAbility(int&, Mercenary\*);

private:

protected:

};

#endif

**Project Code: Grenadier Class CPP File “grenadier.cpp”**

#include "Grenadier.h"

Grenadier::Grenadier(){

stats.attack = 0;

stats.defense = 0;

stats.name = "merc";

stats.type = "Grenadier";

stats.stable = true;

}

Grenadier::Grenadier(std::string n){

stats.attack = 0;

stats.defense = 0;

stats.name = n;

stats.type = "Grenadier";

stats.stable = true;

abilityUsed = false;

}

Grenadier::Grenadier(int a, int b){

stats.attack = a;

stats.defense = b;

stats.name = "merc";

stats.type = "Grenadier";

stats.stable = true;

}

Grenadier::Grenadier(int a, int b, std::string n){

stats.attack = a;

stats.defense = b;

stats.name = n;

stats.type = "Grenadier";

stats.stable = true;

}

Grenadier::Grenadier(int a, int b, std::string n, bool s){

stats.attack = a;

stats.defense = b;

stats.name = n;

stats.type = "Grenadier";

stats.stable = s;

}

bool Grenadier::useAbility(int &number, Mercenary\* mercs){

bool tempBool = abilityUsed;

cout<< "Ability being used.." <<endl<<endl;

if(abilityUsed == true){

return false;

}

else{

if(number == 1)

number--;

else

number -= 2;

abilityUsed = false;

}

return tempBool;

}

Grenadier::~Grenadier(){}

**Project Code: Medic Class Header File “medic.h”**

#ifndef Medic\_H\_

#define Medic\_H\_

#include "Mercenary.h"

#include <iostream>

#include <string>

class Medic : public Mercenary //inherited medic mercenary class

{

public:

Medic();

Medic(std::string);

Medic(int a, int b);

Medic(int a, int b, std::string n);

Medic(int a, int b, std::string n, bool s);

~Medic();

bool useAbility(int&, Mercenary\*);

private:

protected:

};

#endif

**Project Code: Medic Class CPP File “medic.cpp”**

#include "Medic.h"

#include <cstdlib>

#include <ctime>

#include <Vector>

Medic::Medic(){

stats.attack = 0;

stats.defense = 0;

stats.name = "merc";

stats.type = "Medic";

stats.stable = true;

}

Medic::Medic(std::string n){

stats.attack = 0;

stats.defense = 0;

stats.name = n;

stats.type = "Medic";

stats.stable = true;

abilityUsed = false;

}

Medic::Medic(int a, int b){

stats.attack = a;

stats.defense = b;

stats.name = "merc";

stats.type = "Medic";

stats.stable = true;

}

Medic::Medic(int a, int b, std::string n){

stats.attack = a;

stats.defense = b;

stats.name = n;

stats.type = "Medic";

stats.stable = true;

}

Medic::Medic(int a, int b, std::string n, bool s){

stats.attack = a;

stats.defense = b;

stats.name = n;

stats.type = "Medic";

stats.stable = s;

}

bool Medic::useAbility(int &number, Mercenary\* mercs){

bool tempBool = abilityUsed;

srand(time(0));

int numGren = Mercenary::platoon.numGren;

int numMedic = Mercenary::platoon.numMedic;

int total = numGren + numMedic;

int chance = rand() % 2;

int lucky = rand() % total;

cout<< "Ability being used..." <<endl<<endl;

if (chance == 1){

if(abilityUsed == true){

return false;

}

else{

abilityUsed = false;

bool fixed = false;

do{

lucky = rand() % total;

if(mercs[lucky].getStats().stable == false){

changeStat(3, mercs[0]);

lucky = true;

cout << mercs[lucky].getName() << " was resuscitated by " << getName();

}

}while(lucky == false);

}

}

return tempBool;

}

Medic::~Medic(){}