**Project 2**

Title

**Warriors’ Extravaganza**

Course

**CSC-17A**

Section

**48130**

Due Date

**December 5, 2014**

Author

**Kevin Vo**

**Introduction**

Title: Warriors’ Extravaganza

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

**Summary**

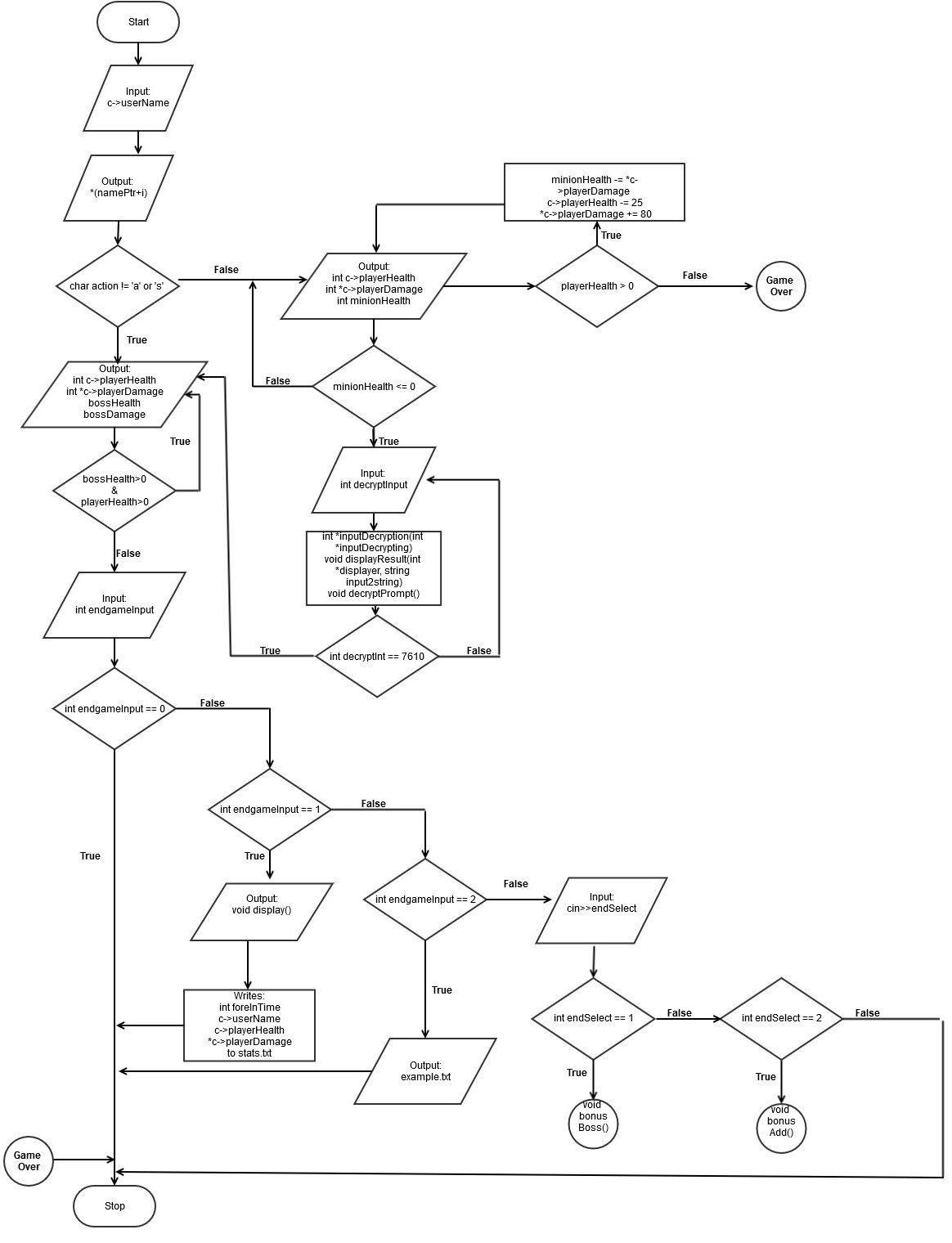
Project Size: 1148 lines

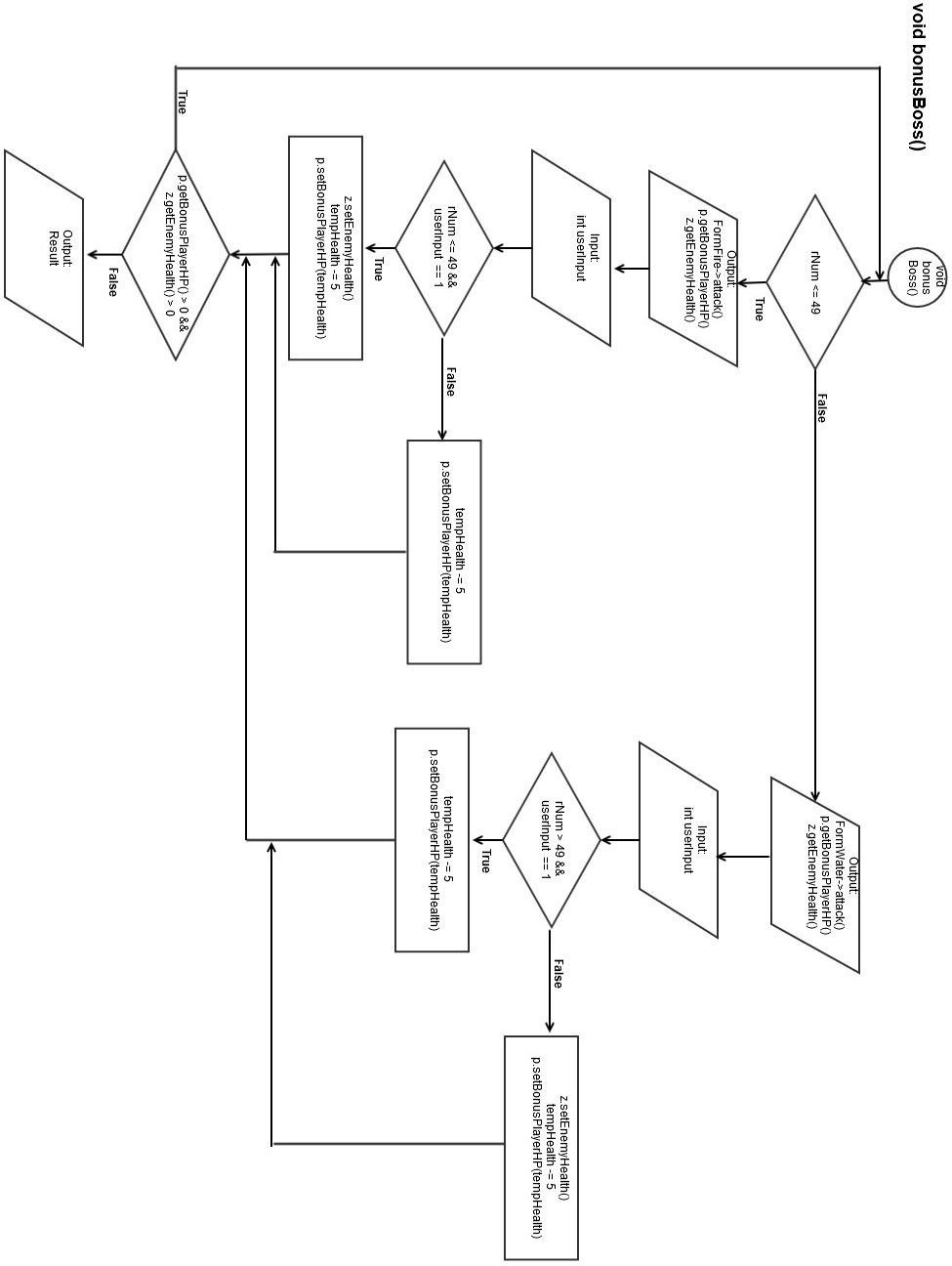
The number of variables: ~30 variables

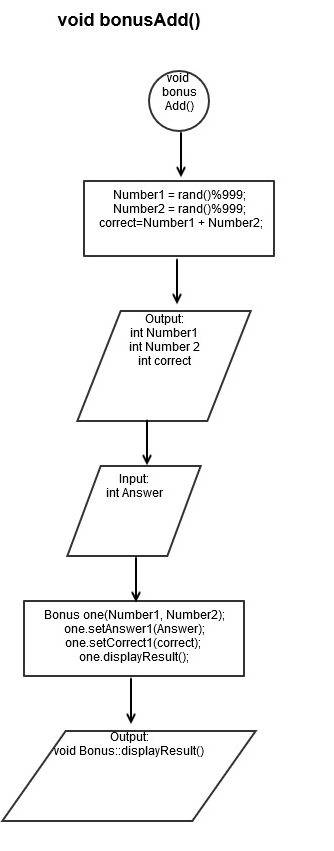
Structures: 1

Classes: 5 (not derived)

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

**Flowchart**

****

****

**Pseudo Code**

*Initialize*

*If enemy1 is chosen*

*Displays status of enemy1 and player*

*If a player button is pressed*

*Player has no health*

*Exit the game*

*Else enemy1 has no health*

*Game continues to enemy1 and/or 2*

*Else if enemy2 is selected*

*Displays code to be decrypted*

*If a player button is pressed*

*Loops until code is correct*

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

*If player has no health*

*Exit the game*

*Else enemy1 has no health*

*Game finishes*

*Display the result*

*If a player button is pressed*

*Status writes and reads a text file*

*Enter bonus menu*

*If a player button is pressed*

*Else*

*Game exits*

**Major Variables**

|  |  |  |  |
| --- | --- | --- | --- |
| **Type** | **Variable** | **Description** | **Location** |
| **Integer** | \*c-> playerDamage | The value of the player’s damage | int combatBoss();  int main();  int minion(); |
|  | c-> playerHealth | The value of the player’s health | int combatBoss();  int main();  int minion(); |
|  | minionHealth | Stores the minions’ health | Minion(); |
|  | bossHealth | Stores the constantly changing value of the boss’s health | combatBoss(); |
|  | bossDamage | Stores the constantly changing value of the boss’s damage | combatBoss(); |
|  | overDose | Holds the number of potions | int minion(); |
|  | \*inputPtr | Pointer array for the encrypted input | int boss();  void decryptPrompt(); |
|  | digitInput | Stores input for decryption | Void decryptPrompt(); |
|  | inputCap (constant) | Makes sure that only four digits are taken for decryption | int boss();  void decryptPrompt();  int \*inputDecryption(); |
|  | endgameInput | Allows player to save data to a text file. | void ReadWrite (); |
|  | foreignHr | Stores 24hr format | class ForeignTime : public Time |
|  | bonusPlayerHP | User’s HP in Bonus | class BonusPlayer |
|  |  |  |  |
| **Char** | action | Allows player to select option at the beginning of game | int minion();  int main(); |
|  | \*namePtr | Points to the array userName | int main(); |
|  | playerInput | Allows the player to selection options during boss fight | Int combatBoss(); |
|  | userName[] | Stores player’s name | struct Play{};  int main(); |
|  |  |  |  |
| **Bool** | btw07 | Makes sure that digitInput is between 0 and 7 digits | void decryptPrompt();  int boss(); |

**Concepts**

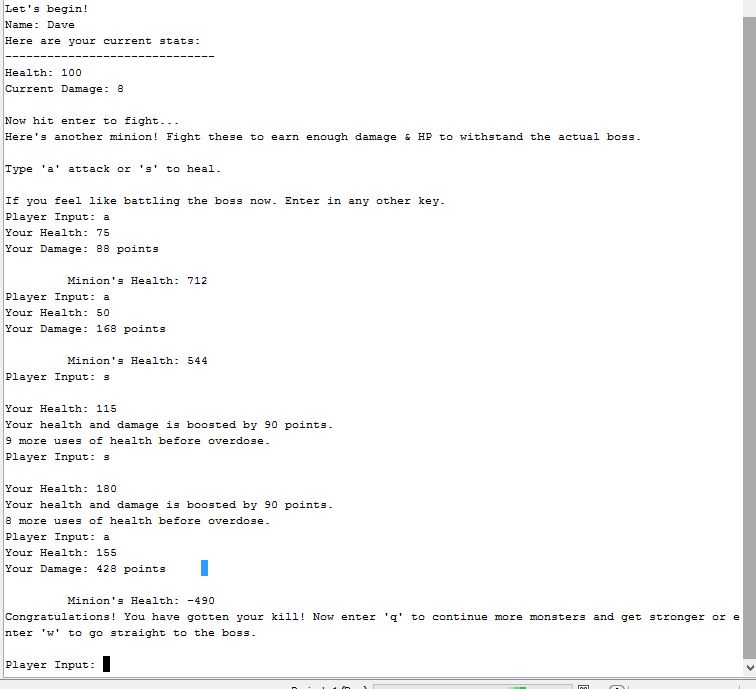
From chapter 9 we learned about pointer variables and their purpose hold the memory addresses of other variables. I used this to hold the address of the player’s inputted username with in my gaming project along with holding the address for the decryption part of the game. For chapter 10 we ventured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

For Project 2, I kept everything from Project 1 including the functions that are not within a class as I see examples of them in the book accepting pointers from structures. With that said, in chapter 13 and 14 I used classes that includes constructors, arrays of objects, defining instances, have mutators and accessors and passing arguments to the constructors and other functions within the class throughout the program. In chapter 15, I used inheritance, polymorphism, and virtual functions. For example inheritance was used to accept and convert time so that the standard format can be displayed as the 24-hour time is copied into a text file. When it comes to polymorphism, I used it along with a virtual function that alters it so that it will be saying two different things with one function depending on the situation. For chapter 16, I used templates with a class for sorting numbers and displaying them so that it simulates wait time by displaying a countdown in a moment of the game.

**Sample Input/Output**

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

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



**References**

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

**Project 2 Checklist:**

1) Dynamic Array -> int sortStat[arrSize]; in Sort.h and char userName[] in

PlayerInfo.h

2) Read & Write -> Lines 46 - 103 in SavePrompt.h

3) Classes -> Bonus.h, BonusBoss.h, Timing.h, Sort.h

4) Inheritance - > class TimeClock:public ForeignTime in Timing.h

5) Polymorphism -> Line 40 and down in BonusBoss.h virtual void attack(){}

6) Template -> template <class T>class SortableVector in Sort.h utilizes a class

**Project 1 Checklist:**

1) Pointers

PlayerInfo.h: int \*playerDamage //a pointer as an element inside a structure

PlayerInfo.h: \*c->playerDamage

PlayerInfo.h \*inputDecrypting

2) Functions with Structures

PlayerInfo.h: int combatBoss(struct Play \*c)

PlayerInfo.h: int minion(char action, struct Play \*c)

3) Pointers with Arrays

main.cpp: namePtr = c->userName;

PlayerInfo.h: inputPtr[count] = (digitInput % 10);

4) Array of Structures

PlayerInfo.h: char userName[10]; (INTERNALLY)

main.cpp: namePtr = c->userName; (EXTERNALLY)

5) Pointer Notation

main.cpp: cout<<\*(namePtr+i)<<…

6) Pointer with Structures

Line 18: int \*playerDamage; (INTERNAL)

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

**Program (not entire program)**

**/\***

**\* File: Vo\_Kevin\_Project\_2\_48130.cpp**

**\* Author: Kevin Vo**

**\* Student ID: 2394221**

**\* Created on December 6, 2014, 6:51 PM**

**\* Line Count: 1148 lines (without the checklist)**

**\* Checklist:**

**\* 1)**

**\*/**

**#include <cstdlib>**

**#include <iostream>**

**#include <ctime>**

**#include <iostream>**

**#include <fstream>**

**#include <cstdio>**

**#include <cmath>**

**#include <iomanip>**

**#include <cstring>**

**using namespace std;**

**#include "Timing.h"**

**#include "Sort.h"**

**#include "PlayerInfo.h"**

**#include "BonusBoss.h"**

**#include "Bonus.h"**

**#include "SavePrompt.h"**

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

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

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

**int dmg = 8, endgameInput;**

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

**c->playerHealth = 100;**

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

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

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

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

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

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

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

**cin.ignore();**

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

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

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

**char \*namePtr;**

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

**namePtr = c->userName;**

**cout<<"\nHello.\n";**

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

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

**cout<<"Name: ";**

**//Pointer Notation example**

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

**cout<<\*(namePtr+i);**

**}**

**cout<<endl;**

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

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

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

**cin.ignore();**

**cin.get();**

**//Calls for minion battle**

**minion(action, c);**

**//Calls for decryption to boss**

**boss();**

**//Calls for boss battle**

**combatBoss(c);**

**//Displays congrats message**

**congratDisplay();**

**//Writes player's status to a .txt or reads previous stats**

**//and displays the endgame options**

**ReadWrite(endgameInput, c);**

**delete namePtr;//cleans up**

**return 0;}**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**//Timing.**

**//Inheritance Example**

**//Gets protected members from Time to convert it and display it**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**class Time{**

**protected:**

**int hour;**

**int min;**

**int sec;**

**public:**

**Time(int h, int m, int s)**

**{ hour = h; min = m; sec = s; }**

**void setTime(int h, int m, int s)**

**{ hour = h; min = m; sec = s; }**

**int getHour()**

**{ return hour; }**

**int getMin()**

**{ return min; }**

**int getSec()**

**{ return sec; }**

**};**

**//Example of inheritence where class ForeignTime is inheriting from class Time**

**class ForeignTime:public Time{**

**private:**

**int foreignHr;//Contains 24hr format**

**int ForSeconds;//Contains secs in standard form**

**public:**

**ForeignTime(int h,int s):Time(0, 0, 0)**

**{ setTime(h, s); }**

**void setTime(int h,int s)**

**{**

**foreignHr = h;**

**ForSeconds = s;**

**Time::setTime(ForeToStandHr(h), ForToStandMin(h),s);**

**}**

**//converts hours**

**int ForeToStandHr(int fore)**

**{**

**fore /= 100;**

**if(fore > 12){**

**return fore - 12;**

**}**

**else{**

**return fore;**

**}**

**}**

**//converts mins**

**int ForToStandMin(int fore){**

**fore %= 100;**

**fore \*= 0.6;**

**return fore;}**

**//Displays the time used foreign countries**

**void getHour(){**

**cout<<"24-hr Time: "<<foreignHr<<" hours.\n";**

**}**

**//Displays the time format used in the U.S.**

**void getStandHr(){**

**cout<<setfill('0')<<"12-hr Standard Time: "<<setw(2)<<Time::getHour()**

**<<":"<<setw(2)<<getMin()<<endl;**

**}**

**};**

**class TimeClock:public ForeignTime{**

**private:**

**int start;**

**int stop;**

**public:**

**TimeClock(int time1,int time2):ForeignTime(abs(time1 - time2), 0)**

**{ start = time1; stop = time2;}**

**void display(){**

**getHour();**

**getStandHr();**

**}**

**};**