Dragons – A simple text based game

This file contains testing results as well as an overview of the software created
COMP2710 Project 1
Mark Gallagher Jr | mag0038@auburn.edu
Filename: project1_results.pdf
Unit testing results obtained using tee command, file created in Microsoft Word. Application created in CentOS 7 on a VMware machine using the Vi text editor and g++ compiler.
Version 4/13/2015

Making the Program

Within the archive is a file named "Makefile". This file holds a script that will 'make' the program's two possible executable files. The following is a description of running each command from the terminal:

- `make` – This is the default command to use for make. This command will automatically make the program's main executable file called "project1_main". This is the production code for the program
- `make project1_test` – This command will make the program's unit testing code. Use this if you'd like to utilize the application's test suite. To do so, run the executable "project1_test" after making. **IMPORTANT**: Ensure that the file "scoreboardTESTread.txt" is in the same directory as the executable. One of the test cases (for loading scores) uses this file.
- `make clean` – This command will run the cleaning script in the makefile. This will delete any file ending with the .o extension in this directory. It will also delete the two possible executables, "project1_main" and "project1_test".

Program Structure

The following is a list of program files and headers used in this application, as well as a basic description for each:

- `project1_main.cpp & project1_main.h` – This file contains the game's main application, or production code. The header file will import all necessary header files and dependencies required for game operation.
- `project1_system.cpp & project1_system.h` – This file contains information that is needed by all other modules of the program. Such data includes certain global constants, enumerated types, and data structures such as the player and action types. The constant defined here set the bounds for the game's attributes. Any type of game balancing can be changed here by altering the values for attribute bounds, time limit bounds, and initial steps away. It also contains some functions that all other modules may use, such as verifying an integer input by the user. Finally, this file will include necessary headers that all functions make use of, such as string, iostream, and cstlib.
- `project1_scoreboard.cpp & project1_scoreboard.h` – This module is used for scoreboard operations. It includes functions for loading a scoreboard from a file, saving a scoreboard to a file, viewing a scoreboard, and adding a new score to the board. The header file defines the scoreNode data structure, a fundamental part of the scoreboard list. It also defines a constant for the default scoreboard file.

- `project1_menu.cpp & project1_menu.h` – This module is responsible for handling the player's choices while in a game. This includes moving forward, reading papers, searching for change, viewing the character stats, and quitting a game. Since this module handles moving forward, it depends on the encounters and puzzle modules.
- `project1_puzzle.cpp & project1_puzzle.h` – This module is used to initialize puzzle presets and ask puzzle questions to the player. The puzzle presets are hardcoded into the program. The possibility of adding the capability to add puzzles from a file was considered, but unfortunately due to time constraints it could not be completed. The module is in charge of randomly selecting a puzzle to ask the player. Then accept the player's answer and determine if they are correct or wrong. After that, the function will update the player's scores accordingly using the points handler available from the System module. This module also defines the puzzleNode structure for the puzzle list.
- `project1_encounters.cpp & project1_encounters.h` – This module is responsible for initializing the encounter dynamic array to preset values, as well as handling any type of encounter the player may experience except for puzzles. This module defines the encounter data structure to determine how to modify a player's attributes. Game balancing can occur here by changing the constants ENCOUNTER_UPPER_BOUND and ENCOUNTER_LOWER_BOUND appropriately. The absolute value of both constants should equal each other.
- `project1_test.cpp & project1_test.h` – This is the unit testing module. All of the unit tests are defined in this module and can be run using make as described earlier. The .cpp file contains a main function which will run through each unit test in sequence until completed. Since this module tests all functions in the application, it depends on all other modules already described.

Other Files Included

There are only 3 other files included in this software package. These files are all .txt files and have to do with scoreboard operations. The first file, "scoreboard.txt", is the default scoreboard storage file. The program will load this file by default, unless the player specifies to use a different file. The other two files, "scoreboardTESTread.txt" and "scoreboardTESTwrite.txt", are used in the unit tests. The read file is used by the load scores test and the write file is used by the save scores test. If running the unit testing module, please do not delete or alter the contents of "scoreboardTESTread.txt" as that will result in an assertion failure and the program will crash.

System-Level Test Results

Table 1 is a copy of the table used in Project 1 – Phase 2. The table is now filled in with results from testing:

Table 1. System-Level test cases and their results

What	Input	Expected Output	Actual Output
Entering a valid username string	"GLaDOS"	Proceeds to ask user if they want to use alt scoreboard	User prompted (y/n) to use alt scoreboard
Entering an invalid username string	"" (empty string)	"Please enter a valid username (longer than 1 character). Try again"	"Please enter your name: " repeats until non-empty entered
Entering a valid system-menu option	2	None. Redirects to appropriate option	Redirected to view_scores
Entering an negative number for menu option	-1	"Option does not exist. Remember to enter 0 or positive numbers only. Try again"	Prompts user again, reminding them to only enter 0 or positive numbers
Entering an out of range menu option	55	"Option does not exist. Try again"	Prompts user again, reminding them that the value they entered was out of range
Entering a letter or string when an integer is needed	GG	"Please enter numbers only, not letters or symbols. Try again"	Prompts user again, reminding them to only enter numbers at the prompt
Entering a valid puzzle answer (correct)	4 (corresponds to answer option 4)	"You are correct!"	Tell user is correct. Updates attributes accordingly
Entering a valid puzzle answer (wrong)	2 (corresponds to answer option 2)	"Your answer is wrong! :("	Tells user is wrong (though no sad face included). Updates attributes accordingly
User enters a valid filename for load/save scores	score1.txt	"Program will load and save scores from file 'score1.txt'"	Program will attempt to load from filename entered. If error in opening occurs, reverts to default file.
User enters nothing for filename	"" (empty string)	"ERROR: No name entered. Program will continue with default name 'scoreboard.txt'"	Program attempts to open file anyways. Since empty string, reports error and proceeds to default file.
File I/O error		"ERROR: Program could not load (save) from file <filename>."	If error in user-provided file, reverts to default. If error in default, reverts to empty list.
A player attribute reaches 0 (game end)		"Game Over"	Reports game over immediately to the

			user. Then explains why the game ended. If victory condition (steps == 0), attempts to save score. Reports if successful or not. Returns to main menu.
Player attribute goes negative		"Game Over" (program should treat it as 0)	Reports game over condition and explains why. Same as results in above test.

Unit Testing Results

The following is the output from the unit testing suite for this application. This data was acquired using the `tee` command. The data was then transferred from the Linux VM to a Windows machine to be copied into Microsoft Word for compilation.

Here is the raw output from the test suite:

```
Unit Test 1: void login_process(string& userName)
      Case 1.1: Enter a valid username when prompted
Please enter your name: Mark G
      Case 1.1 Passed...

Unit Test 2: void init_character(string userName, player& userPlayer)
      Case 2.1: Initialize character to legal values with a given
username
      Case 2.1 Passed...

Unit Test 3: void create_puzzle_list(puzzleHead p_head)
      Case 3.1: Initialize puzzle list properly
      Case 3.1 Passed...

Unit Test 4: void init_encounters(encounterPtr& encArray)
      Case 4.1: Properly initialize encounters array
      Case 4.1 Passed...

Unit Test 5: void view_scores(scoreHead sBoard)
      Case 5.1: Should display 2 scores from list with termination line
The top 2 high scores are:
1. GLaDOS: 9001
2. Chell: 10
-No More Scores to Show-
      Case 5.1 Passed...

Unit Test 6: u_short load_scores(scoreHead& sBoard, const char
filename[])
      Case 6.1: Attempt to load scores from a file that doesn't exist
      Case 6.1 Passed...
```

```
      Case 6.2: Attempt to load scores from a real file
scoreboardTESTread.txt
      Case 6.2 Passed...

Unit Test 7: u_short save_scores(scoreHead sBoard, const char
filename[])
      Case 7.1: Save scores to a file called scoreboardTESTwrite.txt
      Case 7.1 Passed...

Unit Test 8: u_short add_new_score(scoreHead& sBoard, int userScore,
string userName)
      Case 8.1: Add new score to an empty scoreboard
      Case 8.1 Passed...
      Case 8.2: Add new score to the end of a scoreboard
      Case 8.2 Passed...
      Case 8.3: Add new score to start of a scoreboard
      Case 8.3 Passed...
      Case 8.4: Add new score to middle of a scoreboard
      Case 8.4 Passed...
      Case 8.5: Attempt to add an invalid score (0)
      Case 8.5 Passed...
      Case 8.6: Attempt to add score to end of list when list is full
      Case 8.6 Passed...
      Case 8.7: Add new score in the middle of a full list
      Case 8.7 Passed...

Unit Test 9: u_short delete_score_tail(scoreHead& sBoard)
      Case 9.1: Delete tail of list with elements in it
      Case 9.1 Passed...
      Case 9.2: Delete the only node left in a list (Single-element
list)
      Case 9.2 Passed...
      Case 9.3: Attempt to delete an empty list
      Case 9.3 Passed...

Unit Test 10: void calculate_score(player& userPlayer)
      Case 10.1: Calculate normal score
      Case 10.1 Passed...
      Case 10.2: Calculate negative score (and fix it to 0, no
negatives please)
      Case 10.2 Passed...

Unit Test 11: u_short valid_int_input(void)
      Case 11.1: Enter anything and program will print what you entered
(as interpreted by the game)
alpha
      Please enter numbers only: Number
      Please enter numbers only: -4
      Please enter 0 or positive numbers only: 2
      You entered: 2
      Case 11.1 Passed...
```

```
Unit Test 12: void view_character(player userPlayer)
     Case 12.1: Displays a typical player info set
You have:

Money: $1
Intelligence: 10
Time left: 5 turns

You are 3 steps from the goal.
     Case 12.1 Passed...

Unit Test 13: void read_paper(player& userPlayer)
     Case 13.1: Player reads a paper, attributes updated
You read some nonsense claiming that you are being forced to play this
game by an AI overlord bent on testing humans forever, that she's
codenamed GLaDOS, she says 2 + 2 = 10 (in BASE 4!), and that the cake
is a lie.
You gain 5 intelligence, but lose a unit of time.
     Case 13.1 Passed...

Unit Test 14: void search_change(player& userPlayer)
     Case 14.1: Player looks for change, attributes updated
You search through the couches of Shelby desperatly for loose change.
You gain $6! But you lose a unit of time.
     Case 14.1 Passed...

Unit Test 15: void move_forward(player& userPlayer, encounterPtr
encArray, puzzleHead p_head)
     Case 15.1: Nothing happens encounter, moves forward 1
You move forward and...
OH MY GOSH, NOTHING HAPPENS...

You LOSE 1 steps left from the goal
You LOSE 1 units of time

     Case 15.1 Passed...
     Case 15.2: Puzzle encounter, a puzzle should appear. Answer
correctly or wrong, does not matter
You move forward and...
It's puzzle time!

What do you do when life gives you lemons (according to Aperture
Science)?
1. Make lemonade
2. Do what Cave Johnson said and make combustible lemons to burn
people's houses down
3. Go to Food Network and become the next Food Network Star...that
really likes lemons

Choose wisely: 2
*** YOU GOT IT RIGHT! Good Job...are you surprised?
You GAIN 5 IQ points
```

```
You LOSE 1 units of time
You LOSE 1 steps left from the goal


     Case 15.2 Passed...
     Case 15.3: Professor encounter, a professor will appear and
modifiy stats randomly
You move forward and...
You see a tall shadowy figure approach you...

You find a helpful professor who provides you insight into your
current situation

The professor takes up quite a bit of time...
You GAIN 1 IQ points
You LOSE 2 units of time
You LOSE 1 steps left from the goal

     Case 15.3 Passed...
     Case 15.4: Grad student encounter, only time will be lost
You move forward and...
You see a short figure approach you...

You run into your primary arch nemesis!!! He says his usual speech
about how not only will he do better than you, but he will also rule
the world, the galaxy, and then the universe. Once he started talking
about space you got really interested though so instead of moving
onward with your quest, you listen to your nemesis. A slow clap for
you, hero
You LOSE 2 units of time
You LOSE 1 steps left from the goal

     Case 15.4 Passed...
     Case 15.5: Grunt work encounter, loses intel and time
You move forward and...
A small elfish figure hops towards you with something in his hand...

It's a grunt, he gives you a creepy smile and proceeds to throw sheets
of paper at you! Each one is something worse and more mediocre of a
task than the last! You scream for help, no, no, NO, PLEASE NO MORE,
NO MORE WORK!!!, NOOOOOOOOOOOOOOOOOOOOOOOOOO

Whelp, that took some time to do, but you got it done...sadly, you get
no compensation
You LOSE 5 IQ points
You LOSE 3 units of time
You LOSE 1 steps left from the goal

     Case 15.5 Passed...
     Case 15.6: Grade papers encounter, gain money but lose time
You move forward and...
A flying box of papers approaches your position!
```

```
You got papers to grade! But it's all for that 1000-level class...so
you don't get paid a lot

Wow, really, you took THAT LONG to grade? Those papers must've been
really tough...hopefully it'll all be worth it right? Maybe the
Princess will have more money than you, then you don't have to
grade...
You GAIN 1 dollars
You LOSE 4 units of time
You LOSE 1 steps left from the goal

     Case 15.6 Passed...

Unit Test 16: u_short generate_encounter_ID(void)
     Case 16.1: Generate valid encounter ID. TEST REPEATS TEN TIMES.
Each ID is printed
ID Generated: 2
ID Generated: 4
ID Generated: 4
ID Generated: 5
ID Generated: 5
ID Generated: 2
ID Generated: 1
ID Generated: 0
ID Generated: 0
ID Generated: 1
     Case 16.1 Passed...

Unit Test 17: void ask_puzzle(player& userPlayer, puzzleHead p_head)
     Case 17.1: Ask various puzzles they should modify player
attributes. TEST REPEATS THREE TIMES
Take a LEAP forward if you win this one... Who is the most evilest,
rage-inducing, OVERALL BADDEST, League of Legends champion ever?
1. Sona
2. The cute and cuddly Teemo
3. URF the manatee
4. Nocturne
5. Really? LoL? Who plays that...

Choose wisely: 2
*** YOU GOT IT RIGHT! Good Job...are you surprised?
You LOSE 1 units of time
You LOSE 3 steps left from the goal

What does 2 + 2 = ? ... IN BASE 4?
1. 0
2. 4
3. 10
4. 16
5. REALLY? MATH? At a time like THIS?!
```

```
Choose wisely: 3
*** YOU GOT IT RIGHT! Good Job...are you surprised?
You GAIN 10 dollars
You LOSE 1 units of time
You LOSE 1 steps left from the goal

The Holy Overlord Urf the Manatee appears and decides to be nice to
you, but first you answer his question: What does his name truly mean?
1. Unruly Rascals Federation
2. Unsubmerged Reptile of Fire
3. Ultra Rapid Fire
4. Used RAM for Free

Choose wisely: 4
*** YOU GOT IT WRONG!? HOW?? Oh well, better luck next time eh?
You LOSE 5 dollars
You LOSE 5 IQ points
You LOSE 1 units of time
You LOSE 1 steps left from the goal

     Case 17.1 Passed...

Unit Test 18: u_short display_sys_menu(void)
     Case 18.1: Display system menu and get user's VALID choice, will
repeat until choice is valid

1) Start a New Game of Shelby and Dragons!
2) View Top 10 High Scores
3) Quit

     Please choose an option:   5
     That option does not exist, please try again


1) Start a New Game of Shelby and Dragons!
2) View Top 10 High Scores
3) Quit

     Please choose an option: -6
     Please enter 0 or positive numbers only: why
     Please enter numbers only: 2
     Case 18.1 Passed...

Unit Test 19: u_short display_game_menu(void)
     Case 19.1: Display game menu and get user's VALID choice, will
repeat until choice is valid

1) Move forward (could be risky...)
2) Read technical papers (boost intelligence, takes time, could be
fun)
3) Search for loose change (boost money, takes time, maybe find a
sword in a chest...)
```

```
4) View character stats
5) Quit this game and return to Main Menu (why would you ever do
that?)

     Please choose an action: 5
     Case 19.1 Passed...
```