Thomas Hollenberg
CS 161
01/19/2017
Assignment #2 Design Document

Problem Statement:
The local middle school would like some text adventure games to keep their students occupied during down time. The school is leaving it up to your skill and good judgment to develop a game. It is up to you what the story and theme is but there are some requirements:

- There must be at least five different paths or solutions to complete the adventure
- There must be an element of chance that would change a user's chosen path

In addition to design, you must implement your program as a C++ program. Here are some implementation requirements:

- There must be an empty line separating each navigation
- The choice system must be based on numbers
- You must use if/else statements and/or switch statements
- You need to use the rand() function to add an element of chance

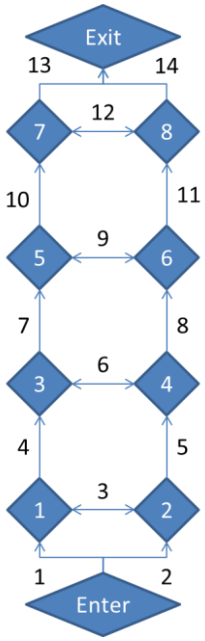
(Extra Credit) Ask the user if he/she wants to play again.

Understanding the Problem:
I need to write a text-based adventure game in C++. The theme and design of the game is largely up to me. However, the game must allow the user to make at least five decisions before ending. Also, there must be an element of chance so that identical user inputs will not always produce the same output.

The events taking place within the game must be narrated by text output to the console. The user must be prompted for numerical input after each game event to influence what happens next. If/then statements or switch statements must be used to control the flow of the program based on user responses as well as random numbers generated using the rand() function. A newline character must be output to the console to visually separate events.

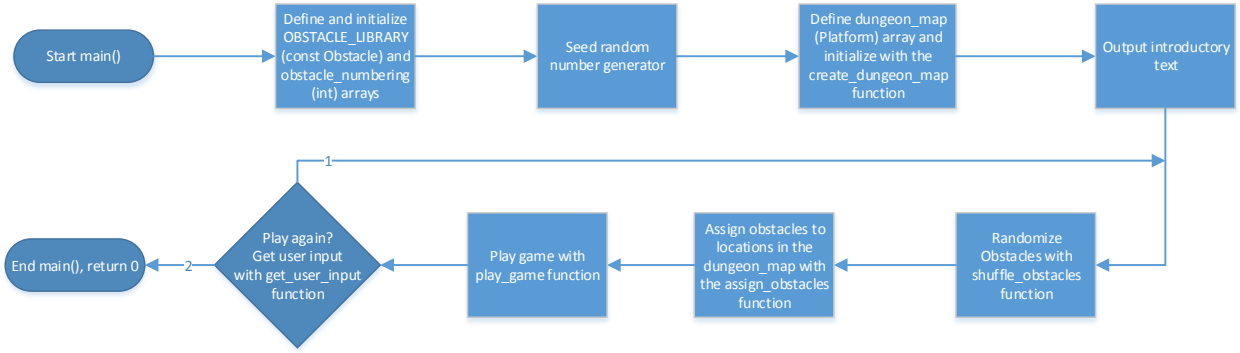
Devising a Plan:
<u>Dungeon Map</u>
Platform (blue diamonds with white numbering):
[Note: "Enter" is Platform 0, and "Exit" is Platform 9.]
- Number (int)
- Straight (Platform pointer)
- Side (Platform pointer)
- Obstacle Straight (const Obstacle pointer)
- Obstacle Side (const Obstacle pointer)

Obstacle (blue arrows with black numbering):
[Note: In code, Obstacle numbers are one less than pictured.]
- Description (string)
- Difficulty (int)
- Prompt text (string)
- Failure text (string)



Each play-through the 14 obstacles will be randomly assigned to the obstacle locations. The user will be asked to choose which obstacle to attempt at each platform. The user can go back and forth laterally, but cannot choose to go back towards the entrance. The user succeeds if a number from 0 to 99 randomly generated with rand() equals or exceeds the difficulty of the chosen obstacle. Failure results in being returned to the entrance, at which point the user will be asked whether they would like to exit the cave (and give up) or keep playing.

main Function



Creating the dungeon map

The create_dungeon_map function will assign a number to each Platform and assign Platforms to the straight and side pointers of each Platform object in the dungeon_map array according to the dungeon map diagram above. (For example, the straight and side pointers of Platform 1 will point to Platforms 3 and 2, respectively.)

Obstacle Randomization

The obstacle randomization technique, performed in the shuffle_obstacles function, will entail iterating through the obstacle_numbering array and, for each of the 14 elements, swapping it with another random element. In the assign_obstacles function, the index of the obstacle in OBSTACLE_LIBRARY assigned to each obstacle location will be equal to the value of the element in obstacle_numbering with an index equal to that of the obstacle location (one less than the black numbers in the dungeon map diagram above).

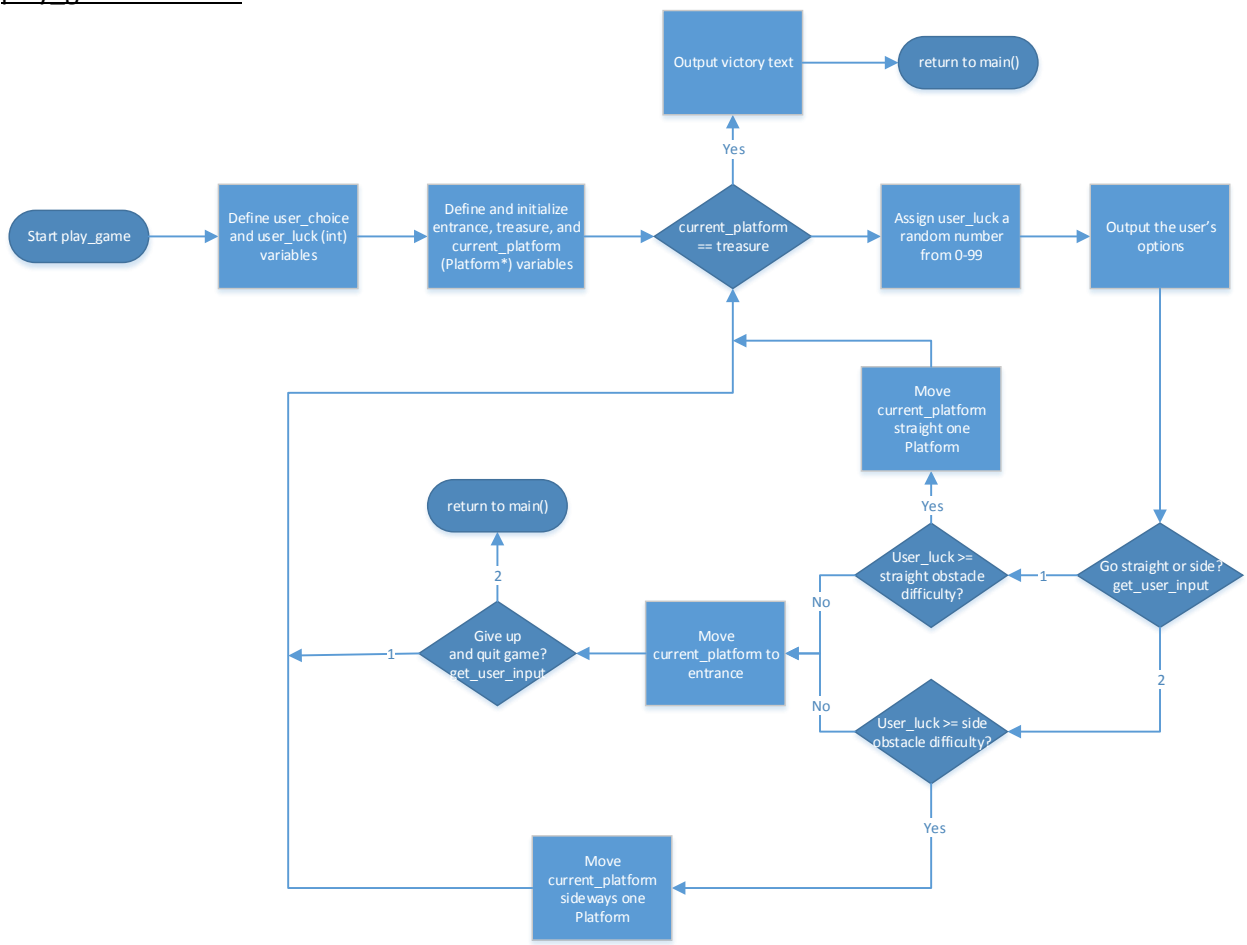
User Input

The get_user_input function will take a string prompt to output to the user repeatedly until either a 1 or a 2 is input.

Introductory/Explanatory Text

You are an intrepid treasure hunter in search of the lost treasure of an ancient civilization of cave dwellers. Finally, you have tracked down the very cave where the treasure is hidden. The cave consists of a series of platforms (some high and some low) above a roaring underground river. Each platform gives access to two other platforms through obstacles of varying difficulty. Choose your path carefully, because if you fail an obstacle, the river brings you all the way back to the cave entrance. Going straight moves you towards the back of the cave, one step closer to the treasure. Moving side to side does not, but allows you to circumnavigate dangerous obstacles you may find in your path.

play_game Function



Predefined set of Obstacles – stored in constant array

1. “a rope hanging from the ceiling. It looks like you could swing across, if you had to.”, 55, “use the rope swing”, “You don’t swing far enough and plunge into the water below.”
2. “a sturdy-looking wooden footbridge.”, 5, “cross the footbridge”, “You stub your toe on a protruding nail and, blinded by pain, stumble into the river.”
3. “a wet log extending over the opening. It looks quite slippery.”, 45, “walk across the log”, “The log is too slippery. You lose your footing and fall in.”
4. “mossy stepping stones spaced a bit farther apart than you’d like.”, 35, “use the stepping stones”, “You make a wrong step and walk right into the water.”
5. “a narrow ledge along the wall of the cave. If you’re careful you could walk across.”, 30, “sidle across the ledge”, “You panic and lose your balance, plummeting into the river below.”
6. “handholds and footholds protruding from the wall. A skilled climber could make it across.”, 50, “climb the rock wall”, “You get stuck halfway across. Your arms fatigue and you have no choice but to let go and dive into the water below.”
7. “a giant eagle beckoning for your to climb onto its back.”, 20, “ride the eagle”, “You accidentally pull out one of its feathers, and the great bird bucks you off into the river.”
8. “a hang glider. There is probably enough room to get a good running start.”, 25, “take the hang glider”, “You don’t pick up enough speed and come up pathetically short.”

9. “a zipline securely fastened at both ends.”, 10, “ride the zipline”, “You lose your grip and plunge into the water.”
10. “a large tree with several long branches extending across the opening.”, 40, “climb the tree branch”, “As you’re inching your way across, you hear a loud crack. Both you and the branch tumble down into the river.”
11. “a narrow gap. You could probably jump across.”, 15, “make the jump”, “It has been longer since your track and field days than you remembered. You take a leap and go right into the river.”
12. “a canoe resting on the shore. Hopefully it doesn’t leak.”, 20, “paddle across in the canoe”, “You hit a rock and the canoe capsizes.”
13. “a set of rusted monkey bars leading across the opening.”, 15, “swing across the monkey bars”, “Halfway through you miss a bar and fall in.”
14. “a slackline stretched across the river.”, 50, “walk across the slackline”, “You lose your balance and plunge into the river below.”

Testing Plan

Testing get_user_input function:

Input Value:	Expected Outcome:	Actual Outcome
1	Returns 1 to caller	Returns 1 to caller
2	Returns 2 to caller	Returns 2 to caller
1 2 1	Returns 1 to caller and ignores the rest of the input	Returns 1 to caller and ignores the rest of the input
1asd	Returns 1 to caller and ignores the rest of the input	Returns 1 to caller and ignores the rest of the input
22	Outputs prompt string again and waits for further input	Outputs prompt string again and waits for further input
/	Outputs prompt string again and waits for further input	Outputs prompt string again and waits for further input
asdf	Outputs prompt string again and waits for further input	Outputs prompt string again and waits for further input

Testing program logic:

Input Value	Expected Outcome	Actual Outcome
Do you take the hang glider (1) or walk across the slackline (2)?		
1	Hang glider success or fail text	Hang glider success or fail text
2	Slackline success or fail text	Slackline success or fail text
Would you like to exit the cave and quit this game (Keep playing: 1, Quit: 2)?		
1	Output scenario narration text and prompt user for next decision	Output scenario narration text and prompt user for next decision
2	Return to main()	Return to main()
Would you like to play again (Play Again: 1, Quit: 2)?		
1	Start next game with re-randomized obstacles	Start next game with re-randomized obstacles
2	End program	End program