Lab 3, C++ STL

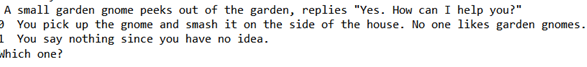
# General Description

Even before Graphical User Interfaces, there were computer games. They were just text based. In 1961, John Burgeson created a baseball simulator on an IBM 1620 mainframe. Text-based games are still being created. The text-based role playing game Open Sorcery was released in 2017 (<https://en.wikipedia.org/wiki/List_of_text-based_computer_games>, 9/5/2019). This lab uses a vector of a user-defined class that has a vector in it to implement a text-based adventure game, and you get to come up with the script.

This application is a text-based adventure game. It’s really more than that – it is actually a game engine since it can read any text file with the game story in it as long as the text file is in the correct format.

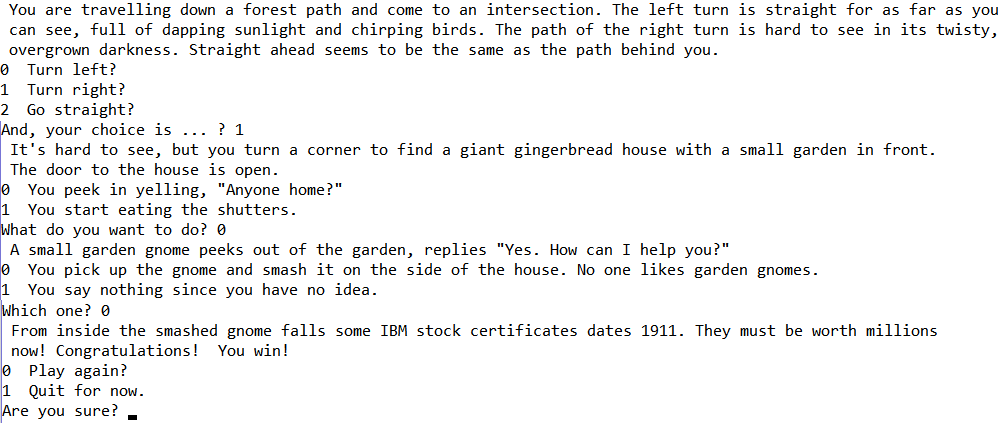
# Detailed Description

Create a project in the IDE called *lastname\_firstname*Lab3 where these are your names such as arnold\_cindyLab3. Inside this project create **at least** the following files with their required data/functions. Your particular implementation of the lab may not use every function listed, but the program must have them.

* prompt.h, prompt.cpp: these files create a Prompt class that implements a prompt such as It's hard to see, but you turn a corner to find a giant gingerbread house with a small garden in front. The door to the house is open. This class has at least:
  + A constant class (static) variable named outputs that is an array of C++ string objects which holds at least ten different phrases such as "What now? ", "And, you pick... :", "What do you want to do? ", etc. These phrases will print after the text of the prompt. See the sample execution.
  + A question, the words in the prompt, a C++ string object.
  + An id, an integer. This number should be the index of the prompt when it is placed in a vector of Prompt objects.
  + A vector of Response objects, responses. For example, the prompt above (“It’s hard to see, but you …) has two responses in the sample execution: You peek in yelling, “Anyone home?” and You start eating the shutters.
  + A default constructor.
  + A constructor that has one parameter, a C++ string object, that is the text of the prompt.
  + A copy constructor
  + Set member functions for the question and the id.
  + Get member functions for the question and the id.
  + An addResponse member function that has one parameter, a Response, and puts that parameter in the vector of responses.
  + A getResponse member function that has one parameter, an index into the vector of Response objects. It returns the single Response object at that index.
  + A clearResponses function that clears (empties) the responses vector.
  + An overloaded insertion operator, <<, that prints out the text of the prompt and its responses. It also randomly selects one of the strings from the outputs array and prints it.  
    
  + An overloaded extraction operator, >>, that reads a prompt from a text input file. See instructions below on creating the text input file that contains all prompt/response data.
* response.h, response.cpp: these files create a Response class that has at least:
  + Private data for a response to a prompt (a C++ string object)
  + An integer that is the index into a vector of prompts of the prompt that should come next after this response. For example, at the beginning of the sample execution the response 1) Turn right? results in the prompt that starts with It’s hard to see, but you turn a corner …
  + A default constructor
  + A constructor that has two parameters, a C++ string object that is the text of the response and an integer that is the index of the response’s prompt in a vector of prompts.
  + A copy constructor
  + Set and get member functions for text of the response and its prompt index
  + An overloaded insertion operator, <<, that prints the text of the response to the screen.
* main.cpp
  + Contains a vector of Prompt objects.
  + Reads game input file into the vector of Prompt objects
  + Plays the game as shown in the sample execution until the user is done using this vector of Prompt objects.

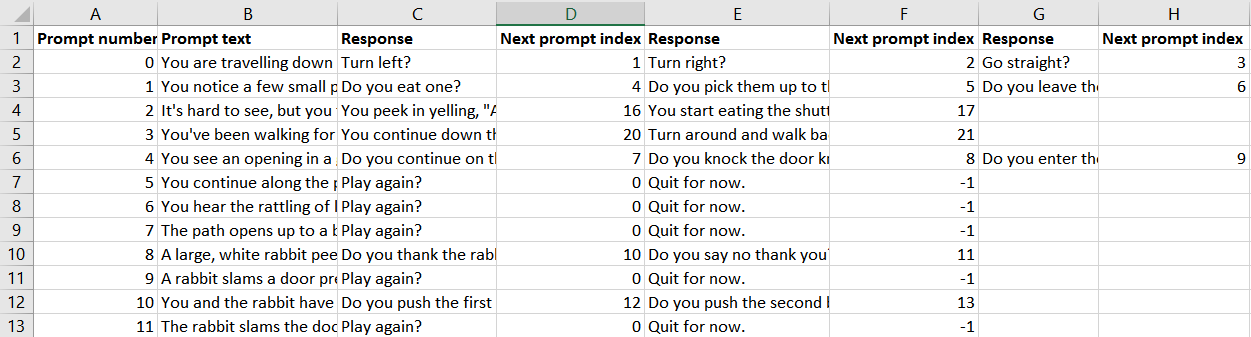
This lab is all about STL vectors. The program MUST use STL vectors where vectors are specified.

## Sample Execution

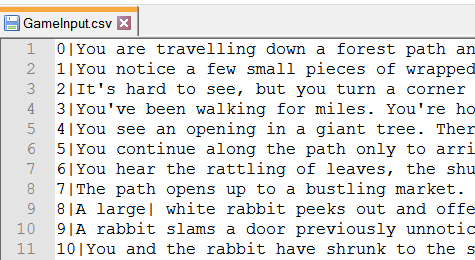


## Creating the Input File

Use Excel to create the input file. For example, see MS Excel file GameInput partially shown below.



Make your own story by creating an Excel file like GameInput. Don’t use commas in any of the text since, when saved as a CSV file, fields will be separated by commas. Remove the column headings and save the Excel file as a comma delimited CSV file. This is just a text file that you can open with Notepad++. Also, the program can read it as a text file. Each field will be separated by a comma. If you want to use commas in the text, replace all commas with another delimiter not used in the text such as a horizontal bar, |, then type the commas in the text. You might also want to add the delimiter after the last field which could make program code to read and parse the data easier. See GameInputFIXED.csv for an example.

****

MAKE YOUR OWN EXCEL INPUT FILE WITH YOUR OWN STORY. YOU MUST HAVE AT LEAST 30 PROMPTS.

**HINT:** To read the data, read each line using *getline* then parse the string using one of the options below.

* C-strings: use the *strtok* function to “tokenize” the C-string (<http://www.tutorialspoint.com/c_standard_library/c_function_strtok.htm>).
* C++ string objects:
  + Use a *stringstream* object and *getline* within the *stringstream* class <https://cplusplus.com/reference/sstream/stringstream/?kw=stringstream>, https://www.tutorialspoint.com/stringstream-in-cplusplus
  + Use *find* and *substr* functions in the string class <http://www.cplusplus.com/reference/string/string/?kw=string>

In a file called README in the same folder as the workspace file, list one path to win and one path to lose so I can test winning and losing without playing the game forever. Also, include the answers to the Relevance questions below.

# Relevance

Relevance questions to be answered in the README file.

1. What is one of your favorite computer/video games and what language(s) was it written in? If you don’t have a favorite game, then just find a game that sounds interesting and answer the question for that game.
2. Is Minecraft a video game? Why or why not?

# Rubric

For any credit on this lab, the program MUST read data from a text file in the format described into a vector of Prompt objects, use this vector to play the game, and include and use the files prompt.h, prompt.cpp, response.h, response.cpp, and main.cpp for any credit.

* (10 points) Comments and style
* (10 points) Relevance questions and instructions on one way to win and one way to lose in a file called README at the top of the IDE’ workspace/project folder hierarchy – in the same place as the workspace file which is not necessarily in the same folder as the code or executable file
* Program correctness
  + (10 points) At least 30 prompts for your own game in a text file in the format described
  + (10points) Prompt class
  + (10 points) Response class
  + (50 points) Correct game play

# What to Turn In

Create a Windows zip file called *lastname\_firstname*Lab3.zip with the entire *lastname\_firstname*Lab3 IDE’s workspace/project folder. This is the folder that contains the workspace/project file. Unzip it in another folder and verify you can use the IDE to run the program without building it. Turn in this zip file in the online assignment.