Technical Report

a. ADTs

Linked Lists:

We will use linked lists to store the questions and answer choices for the game. We will have six text files corresponding to different question categories (which correspond to button colors on the GUI), and will load these into six linked lists. When a player selects a button of a certain color on the GUI, questions from the corresponding linked list will be called.

We chose to use linked lists to store the questions because we are loading questions and answers from text files of unknown size, and also because we want to sort of randomize the order in which questions are asked, and a queue or stack would mandate that questions are always presented in the same order (taking items from the top or bottom of the stack or queue).

Graph:

We will use a modified version of the adjacency matrix graph implementation to keep track of which buttons on the GUI have been clicked and where a player is allowed to move (in the version of the game where they roll the dice). Each vertex on the adjacency matrix will correspond to a button on the GUI.

For the version of the game which involves rolling a dice (where the winner is the first to answer a question for each category), the adjacency matrix will be used to determine legal moves. We will add a method to the AdjMatGraphPlus class to determine the distance between vertices, and this will determine where a user is allowed to go based on their dice roll.

For the version of the game which does not involve rolling a dice (where the winner is the user who answers the most questions correctly is the winner), the users must attempt a question for each square (button) on the game board, so when a button is pressed, the vertex corresponding to the square will be removed from the AdjMatGraph. When no vertices remain, the game is over.

We chose to use a graph because our gameplay depends on either paths or the number of unclicked buttons, which is best represented by a graph.

b. Classes

TrivialPursuitGUI:

This class will create the GUI using the files corresponding to the tabbed panes.

AboutPanel:

This class will create the panel that contains instructions for the game.

GamePanel:

This class will contain the code that allows the user to use the GUI to play the game. This class uses code from the TrivialPursuit class.

TrivialPursuit:

This class will include code that is used by the GamePanel and encapsulates the rules of the game.

TPQuestions:

This file loads the game questions from text files and stores them in six linked lists.

TPGraph extends AdjMatGraphPlus:

This file will keep track of the game board and will allow TrivialPursuit to determine if a move is legal (for the version of the game using dice) or if the game is over (for the version of the game not using dice). This class will be identical to AdjMatGraphPlus, but will include two additional methods.

c. Main actions of classes

TrivialPursuitGUI, AboutPanel, GamePanel:

These classes set up the GUI and respond to user actions.

GamePanel:

This class will set up the game and will respond to user actions. The clickable buttons on the GamePanel of the GUI will include a Start New Game button and a button for each square on the game board.

TrivialPursuit:

This class will include methods to call questions corresponding to a specific category, keep track of player names and scores, determine if a game is over, and determine the winner of a game.

TPQuestions:

This class will include a method to read information from a text file into a linked list.

TPGraph extends AdjMatGraphPlus:

This class will be identical to AdjMatGraphPlus, but will include one additional method, which will determine the number of steps between each vertex. This will allow us to determine if a user’s move is legal based on their dice roll.