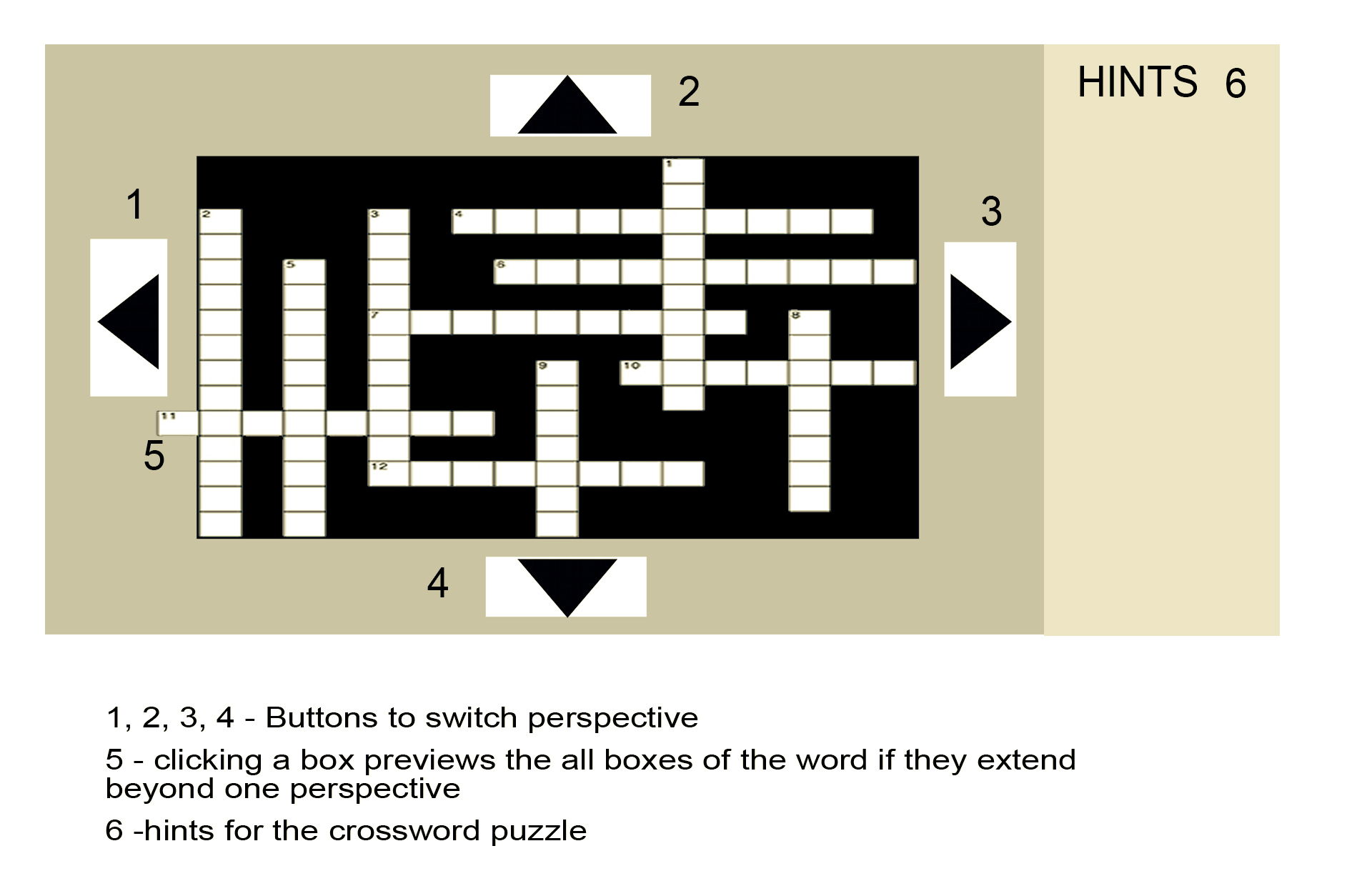
Pedro B. Balduino Morais, Arnav Parikh, Drake Kufwafwa, Harris Khawar

**Part 1 -**

**What will your program look like at the end?**

Our program will operate on its own window, and will initially have some UI for the user to specify a file containing a set of words and definitions to be used to generate a crossword puzzle. The program will then generate a 3D crossword puzzle using the list and display it graphically for the user. The “3D” puzzle will consist of five 2D faces that are mapped onto a cube, with the puzzle extending between faces. The user will then be able to switch between these faces by clicking on on-screen buttons, will select rows/columns with the mouse and will type with the keyboard to complete the puzzle. The crossword key and controls will be displayed to the right of the puzzle. Below is a diagram of what we conceive the program will look like:



**How will the user input work? (e.g., clicking, keys on the keyboard, specifying a file, what have you)**

Firstly, the user will specify a .txt file in a specific format that contains words and their hints/definitions. After the program generates the crossword, the main user input will be clicking - they will be able to click on the puzzle to select specific columns/rows, and on on-screen buttons to switch the current face of the cube. Finally, the user will be able to use keyboard keys to type in the words after a row/column is selected.

**How will the program respond?**

*User specifies file on input:*

The program will respond to receiving a specific file by parsing it into the separate words and their respective definitions, and then storing these as some sort of data.

*User clicks on switch perspective button:*

The program will rotate the cube so that the user is exposed to the selected perspective.

*User clicks on array:*

The program will highlight the array graphically, displaying it in full if it extends onto another cube. The program will also specify that array as the one currently selected by the user in the code, so that keyboard input affects that array while it is selected.

*User types keys while array is selected:*

Typing keys while there is still space in the array will store the character value of that key in the next available spot in the array. Pressing the backspace key will cause the last character in the array to be turned into “null”. Graphically, the program will only display characters that are not “null”.

**What purpose does it serve? (e.g., is a game, a productivity tool, a screen saver?)**

The program is a game. It is intended to be used for the sheer excitement of solving pseudo-3D crossword puzzles generated by yourself or by your friends, or even from word lists found online.

**Part 2 -**

**CLASS PUZZLE:**

This will be the “main” class that will actually run the program. As such, there won’t be a specific instance of the class when the program is run; it simply provides the structure for the program. This is the class that will hold all instances of the other class. It will also extend JFrame to provide the new window and graphics functionality.

**Fields:**

* instanceOfGrid that holds the structure of the grid and the instances of the Word class. We have not yet decided how to hold all the words, whether we will use a 2D array, other data structures or a data structure defined by ourselves.
* private char[ ] currentArray; *//array currently selected by user*
  + This field is private because we do not want to use it in code. We only want it to change in response to user input.
* someStructure currentFace - a sub-structure of the someField class, that will hold the grid for a specific Face of the cube. This field will store the face currently viewed by the user.

**Constructor:**

* The constructor will take in the .txt file inputted into the program. It will have some functionality to detect if the file is in the right format. It will then call parse() on the file.

**Methods:**

* void userClicks(coordinates for click location) { *//processes the user click*}
  + If user clicks on perspective button, call switchPerspective() with respective button name (LEFT, RIGHT, UP or DOWN).
  + Else, if user clicks on array, call chooseArray().
  + Else, set currentArray = null.
  + Should be static because we will not have a specific instance of the puzzle class.
* void switchPerspective(String buttonName) {*//switches the current perspective of the user*}
  + Set current face to a new face depending on whether “LEFT”, “RIGHT”, “UP” or “DOWN” is the input.
  + Should be static because we will not have a specific instance of the puzzle class.
* void chooseArray(char[ ] newArray) {*//sets current array to new input array*}
  + Sets currentArray to new input array
  + Should be static because we will not have a specific instance of the puzzle class.
* void keyPressed() {*//processes when user presses a key*}
  + If currentArray is not null, story key pressed on next available character or delete last character if key is backspace
  + Otherwise do nothing
  + Should be static because we will not have a specific instance of the puzzle class.
* void paint()
  + This method will handle the graphics of the program, and should be called whenever they need to be updated.

**CLASS WORD:**

This class word represents the individual words that were parsed from the .txt file and are supposed to be used in the program. There will be several instances of this class while the program is being run, with each instance being one of the words used in the crossword puzzle.

**Fields:**

* char[ ] wordChars
  + char[ ] that contains the chars that make up the word
* char[ ] userArray
  + char[ ] that contains the chars inputted by the user(this is the array that will be displayed graphically in the final program)
* String hint
  + the hint that corresponds to the word this specific instance holds

**Constructor:** public Word (String word) {...}

* Argument is a string word parsed from .txt file
* Converts the word into a char array and stores it into wordChars.
* Creates a userArray with length equal to wordChars.length.

**CLASS GRID:**

This class will represent the structure that holds the words of the puzzle, and will contain the actual formatting of the puzzle. We have not yet decided how we will organize the data structure, but we want it to contain the following information:

* All specific instances of the Word class
* Some “grid” or structure that stores the connection between the words in the puzzle, and the location of each word

**CLASS GENERATOR:**

Class responsible for generating the structure of the crossword puzzle

**Fields:**

* listOfWords
  + Some data structure that will hold the list of words/hints, allowing to easily get a word from its hint and vice-versa.

**Method:**

* void parse(.txt file)
  + parse a .txt file into words and hints, storing them into instances of the Word class
* generate()
  + This will be an algorithm for creating a crossword puzzle from a list of words.
  + This method will return whatever we decide the class Grid will end up being. Therefore, it will return the structure for the puzzle created using that specific list of words.

**Basic structure of the program:**

Class Puzzle will hold one specific instance of the Grid class, which represents the basic structure of the puzzle. The Grid class will contain the structures that represent each of the 5 faces. Each of these structures will then contain several instances of the Word class, corresponding to the words being used for each specific puzzle.

Class Generator is a helper class that is used within Puzzle to generate the specific instance of Grid corresponding to the list of words inputted.