Project: Digital crossword generator

Analysis

Problem Identification

As young people develop and gain more freedoms it becomes very easy for them to ignore or simply forget about very important activities which will benefit them now and in the long run. The main premise I'm talking about is puzzles and puzzle games which are overshadowed by more flashy pure video games. There's nothing wrong with these games, if anything it's a healthy way to destress, socialise and just have fun every once in a while.

I asked classmates about their thoughts on this and the majority agreed that very little to none of their online time was spent completing puzzles. Most weren't even opposed to doing any; they said that the resources online were lacking whether it be due to presentation or content.

The problem lies in that due to the lack of sources not enough people are making use of the internet for brain training. People naturally flock to the easiest option so to counteract this there must be an accessible, quick to pick up digital puzzle of some sort. It would also need to have enough features and be fun or challenging enough to keep the users coming back. Mental health studies show that… My proposal is an online crossword generator which utilises technology to provide a modern experience to the classic puzzle.

The hardware needed for this solution would likely be very low spec depending on the optimisation. The advantage of a computational solution is to make it able to randomly generate different crosswords and to essentially provide more content than any newspaper/book can give. This comes alongside the fact it is easier to implement accessibility and quality of life features.

Stakeholders

Although my solution can be utilised and enjoyed by anyone it is primarily targeted towards those whose brains are still developing and who have a little more independence than children. With this in mind I have decided my target audience to be teenagers of various ages and have selected a couple of my fellow classmates to act as my stakeholders.

One of the main driving points of my solution is to make it as accessible as possible to those with mental or physical disabilities. To add the necessary features and toggles I will analyse online chat forums and search up suitable standards to implement.

Why is it suited to a computational solution?

The premise of my solution is to be a random crossword generator and it cannot generate any crosswords without a computational algorithm working behind the scenes. It is quite literally impossible to do this without computer systems. The only alternative would be to have a person or group of people manually create these crosswords which would be slower, potentially quite expensive and less tailorable to individual demands.

A key driving point of my solution is accessibility options which computers excel at providing with the vast amount of togglable settings which can be implemented.
Finally, because there are so many words and their definitions, storing them in databases makes for extremely efficient lookup and can be fed straight into the crosswords. Alongside this specific user data must be stored in files.

Computational methods used to solve the solution:

Problem Decomposition:

The problem can be broken into a set of much smaller problems to solve, roughly written as:

-Gathering a list of words and their definitions
-Randomly linking these words together by intersecting letters
-Formatting this into a crossword format

-Producing a ui for this which takes in user inputs

Most functions and procedures will be written standalone with realistic test data and will be combined together to form a working solution.

Abstraction:

Parts of the problem can be disregarded in order to focus on the main task. This will be mostly utilised when producing the crossword grid where the words' definitions will be ignored. It will also be utilised while creating the user interface, particularly in making a static design first and then adding links to the buttons to bring it all together.

Parts of the problem can be disregarded to focus on the main task. This will mostly be utilised when creating the crossword grid when the definitions will be ignored. UI will also benefit from abstraction as a static design will be prioritised first and details will be implemented later.

Interview

The goal of the interview is to gather information and specific feedback from the selected stakeholders. There is a mix of general and more detailed questions in order to tick all the bases

Interview Questions

1. Have you ever played any digital problem solving games? If yes what was it
2. Do you agree with the statement that in the current age it is easier and more accessible to access resources online?
3. What're your thoughts on there being an online crossword generator to keep people's brains active
4. If there were a resource like this, what features do you think it would need
5. Anything else you would like to add on the subject

Question 1 is there to see what experience they have with digital problem solving games in order to test whether my original hypothesis of the internet being underused is correct. Question 2 further delves into this topic

Question 3 proposes my solution and checks to see if they see it fit to solving the problem. Question 4 gathers various ideas from the stakeholders which would increase the quality of the solution. Anything listed here will be considered for development when reviewing the responses. Question 5 gives a final chance to expand on any of these.

Interview

Student A

1. I've played a few Zelda games and Wordle if that counts. I sometimes find myself doing online quizes and stuff for fun although the majority are badly done.
2. Yes
3. I've never thought of that but it sounds like a good way to get young people thinking.
4. A crossword, difficulty levels, competitive features, skins and colour design, especially for the younger audience. Maybe make them unlockable by progression so that people have a reason to keep playing .
5. Other than it sounding like a good idea not really

Student B

1. I play Tetris occasionally but I've enjoyed every time we've had a crossword or word search in school.
2. Yes but it can be overwhelming depending on what is presented to the user.
3. Sounds like a good idea if it can be made fun.
4. Difficulty settings, skins, daily challenges. Background music also so that people can zone out and relax. You could do what other games do and record people's stats so that they can view them.
5. No.

Analysis

The majority of my stakeholders agree with the idea of having a modernised crossword generator to keep people thinking actively. They also agree that there is a lack of exposure for such applications targeted towards students on the market

On the subject of necessary features it seemed agreed upon that difficulty options add more depth to the solution which I agree with and would like to implement.

They also suggested skins and customisation options to keep the younger audience enticed. I particularly like Student A's suggestion on making it unlockable by progression: a tactic many games and applications utilise to increase retention and keep the content fresh.

Student B brought up a stat tracking idea which I liked. I asked them further on it and what content they think it should record. The following is their response:

"Crosswords completed, words answered, time taken and more. Recording these separately for each difficulty would add more depth"

I agree with this statement and the features listed. I will implement these and potentially add some more such as letters answered.

Research

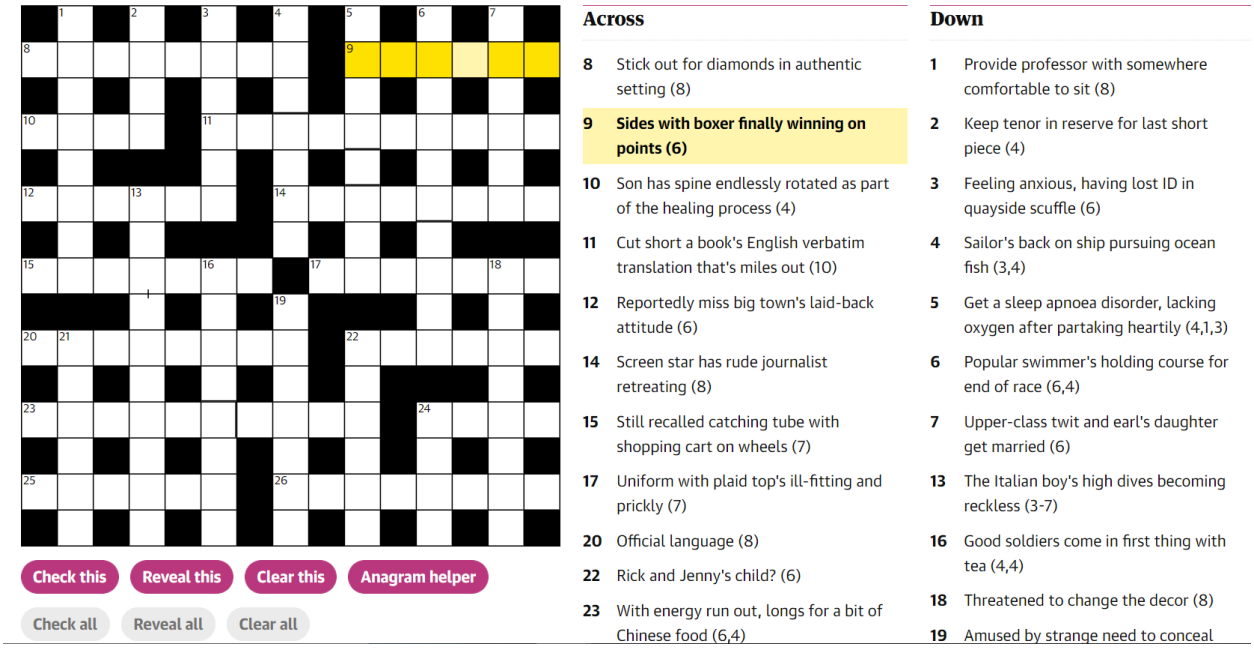
Existing solutions

The Guardian Puzzles:

The main competition for an online crossword system is The Guardian's Puzzle App/ Webpage. They take the crosswords used in their papers and post them up online for anyone to do. Their solution does both good and bad things.

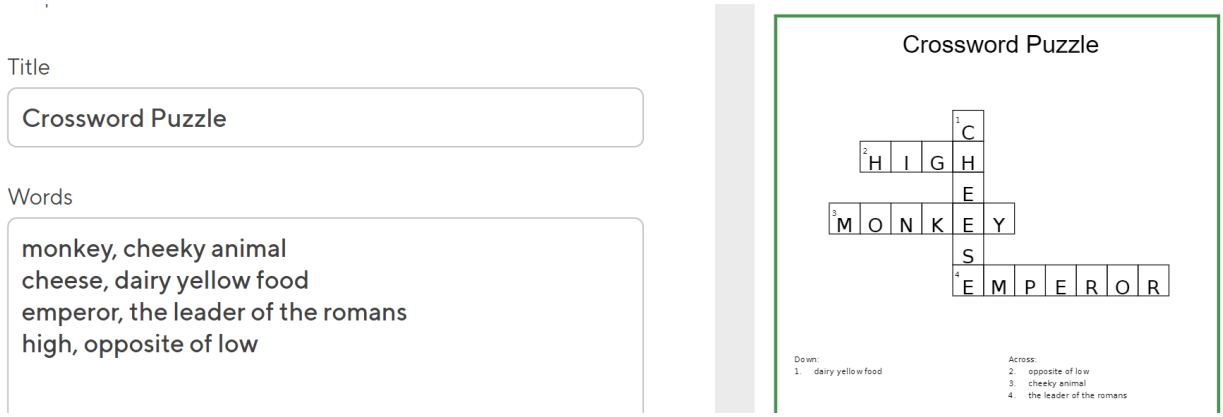
One of my immediate reactions was how clustered the homepage looks

There is an overload of information which is what I want to avoid when catering to my stakeholders of generally less technical people. I do think the different crossword types featured are good but not necessarily front page worthy.



The above screenshot is one when you are doing a crossword. This is where The Guardian stands out as they have clearly separated Across and Down sections, a simplistic readable grid and solutions/ a checking system to help when attempting it. There's also the slight detail of having alternating white and very light grey cells which really adds to the visibility and style of the grid. In my solution I'll reuse these ideas as I see very few ways to improve upon them. I also really like the way the numbers highlight both on the crossword and the questions side. It greatly improves clarity and I'll prioritise adding this to my solution.

Education.com



Above is a screenshot of the website education.com ‘s crossword maker. It takes in words from the user and creates a crossword from them. I don’t think I will implement
Custom crosswords like this as it takes large effort in the user inputs and doesn’t bring enough benefits to the table to warrant potential extra clutter. The concept of adding your own definitions might be good for very niche situations but for the time being I’ll only add preset dictionaries as it simplifies the user experience and depending on the quality of them enhances it.

The presentation of this grid also highlights how much of an impact it actually has on the user. The small details such as letters being aligned and well positioned answer sheets are shown to have a large impact based on this comparison.

Conclusion

From all the research I did I found no actual crossword generators online, the closest being the crossword creator in figure 2 which requires massive effort upon the user’s end. Despite this both sites do present crosswords to the user

Despite this both existing solutions do present crosswords to the user and certain features used by them can be borrowed and expanded upon, The Guardian website in particular.

Features of the proposed solution:

Initial concept of my solution considering this research:

My solution will be an application which when started will take the user to a simplistic interface where they will be presented with the option to generate a crossword, change accessibility settings, and a miscellaneous option. Upon selecting and generating a crossword they will be taken to another screen with the difficulties laid out and explained upon hovering over them. When selecting one of these the application will generate a crossword based on pre-existing dictionaries and present it to the user alongside the numbers and definitions to the side. There will be a verify button for the user to check their progress and the crossword will present a victory screen when completed properly.

Based on the research and interviews I did I believe a difficulty system is necessary for user engagement and retention due to its progressive overload. It would also widen the range of potential users, being either familiar with similar applications or brand new to the scene. This would include a minimum of 4 toggleable options, each building upon the potential vocabulary of the last.

To increase user retention and add more depth to the solution progressive features such as skins and unlockable difficulties will be implemented. There will be further questionnaires sent out in order to find out what my target audience wants, particularly with skins and music. For the unlockable difficulties the fourth difficulty, CHIMPS, will require the user to complete 3 hard crosswords before being able to play.

In terms of accessibility there will be a colourblind mode for most common variants to cater to that audience; I can use online standards to tailor this. There will also be text size adjustment to make it easier to see small text for the visually impaired as well as high contrast mode. By default the text will be set to Comic Sans to help dyslexic people however this can be disabled.

It was also recommended to me to record data after the user has completed a crossword and make it available to them on a different page. This will require a stat tracking algorithm which executes once a crossword is completed and also a records page to actually view this data within the application.

My research also concluded that there is an audience for assistance options such as a hint button. My solution will take this in and implement a hint button.

Limitations of my solution:

The main limitation of my solution is catering to every disability out there, particularly the lack of inbuilt speech to text support which I will be able to give although there may be third party workarounds users may be able to use.

Requirements:

Software and hardware requirements

Hardware

Recommended system requirements (based off of testing):

OS: Windows 10 (64bit)
Processor: Intel i5
Memory: 4 GB RAM
Storage: 20MB
Graphics: Intel HD Graphics 4400 (integrated)

This is in line with making a compact easy to run solution which makes it available to more potential users.

Stakeholder requirements:

Design:

Easy to understand label and button names - Especially important for the target demographic
Difficulty options
User tutorial - Teaches users how to use the features
Skins - Attract more users and increase retention. Also look nice

Functionality

Generates crossword upon input
Algorithm to check if user has solved the crossword
Difficulty features - changes which words appear
Settings menu - toggle certain accessibility options etc

Stattracking - record user’s data from previous crosswords

Success Criteria

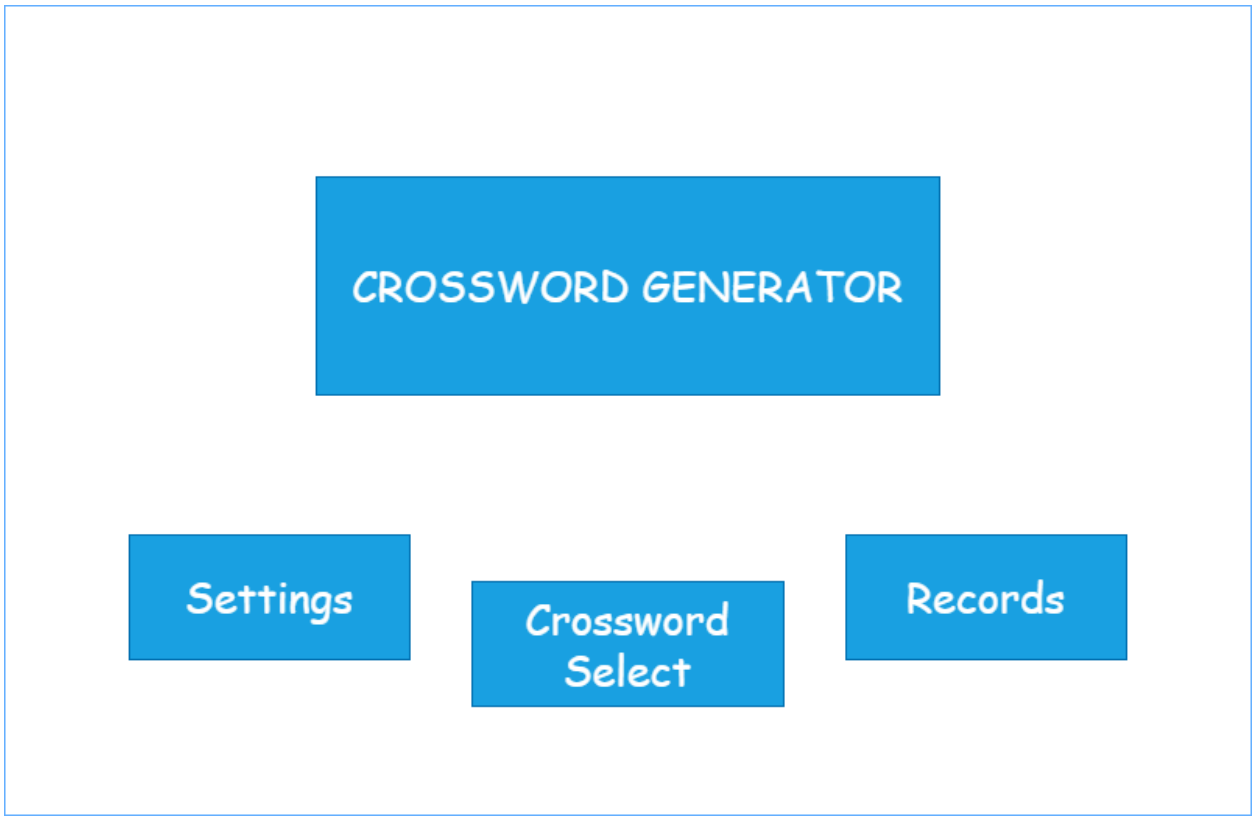
The success criteria for my solution must be specific throughout meaning there is no ambiguity for results. In cases where performance is being measured the criteria must be measurable by statistics either from the solution or feedback from testers. It must be achievable and not overly ambitious. Given the target audience of my project it is fairly realistic to expect the users to be able to type in answers and

Objective	**Criteria**	**How to evidence**
Home page	Home page which navigates to the settings menu, difficulty select and records page	Screenshots
Difficulty page	Difficulty select which displays the difficulty options. Pressing one of the buttons sets variable difficulty to the one selected. User is then directed to the crossword page.	
	Crossword is created of words of the difficulty selected	Screenshots of different difficulty’s crosswords
Stattracking system	A records system which records time taken, average time, words answered, letters answered and crosswords completed for each difficulty.	Screenshots of records page and evidence that it updates successfully after completing a crossword
Records page	Records Page which displays the users stats recorded by the records system.	Screenshots
Crossword page	Crossword page which displays the crossword, word clues and input box.	Screenshots
Down and across clues	Word clues allow user to select which word to answer on the grid by pressing on one	Screenshots

Input Box	An input box which takes user input and depending on which clue is selected adds it to the grid	
Highlight grid	When a clue is selected its position is highlighted on the grid	Screenshots
Hint button	Hint button which reveals a letter on the grid after pressed by the user	Screenshots
Verify button	Verify button which checks the user's answers and displays the Victory Screen if they're correct. If wrong will flash red	Screenshots
Victory screen	Victory Screen which displays after Verify. Displays time taken, words answered and letters answered for the crossword which was just completed	Screenshots
Settings page	A settings page with functioning toggles for: Colourblind mode,high contrast mode, Large text and Font type.	Screenshots of the screen after each toggle is applied
	Functioning skins (colour palette changes) which can also be selected within the Settings Page. Only one of these can be selected at a time	Screenshots of each skin applied to the program and of the program not allowing 2 at once.
Tutorial	An informative and easy to understand tutorial viewable on Settings Page. Tells the user about my solution's features and how to use them.	Screenshots

Design

User interface design:



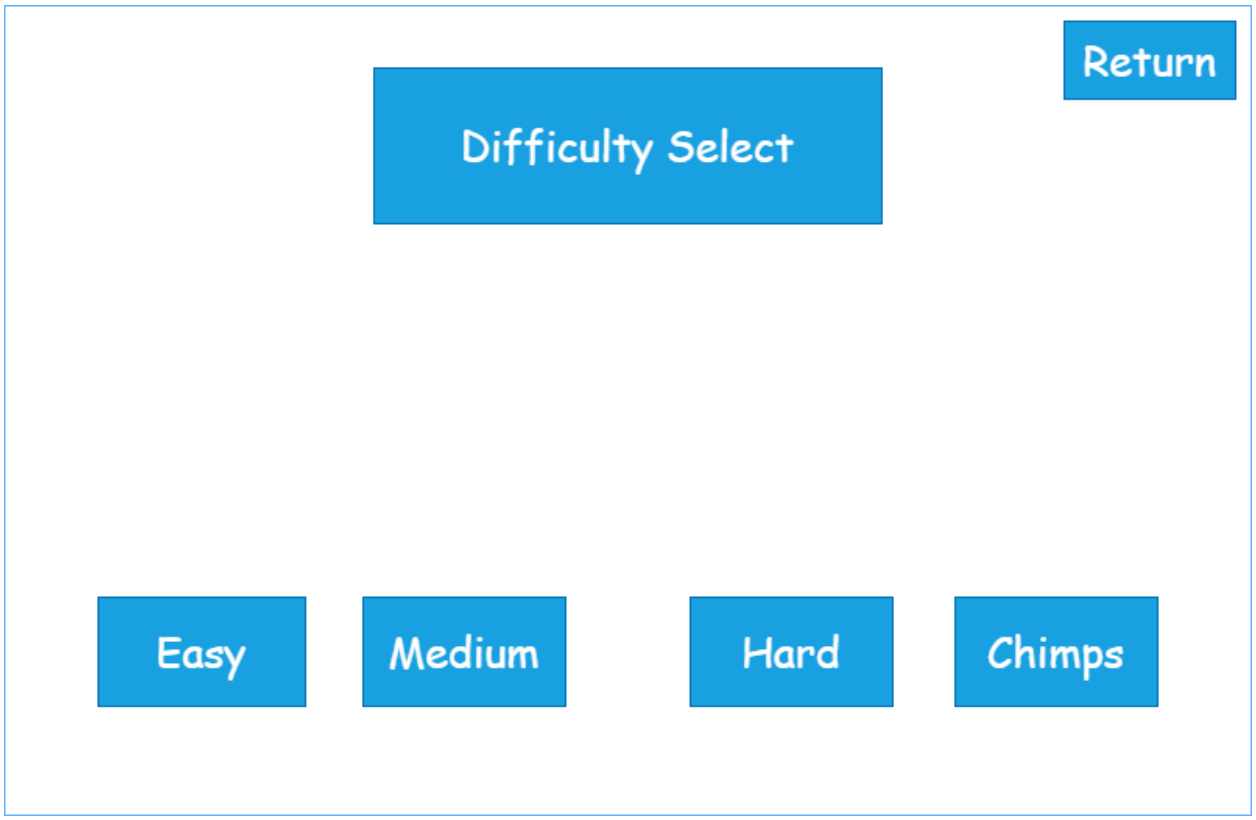
This is the homepage design, the first thing users see when they open the application. It is simplistic and easy to navigate with extra info provided by hovering over buttons after a second. Accessibility is easily accessible through here so that first time users can sort their settings out before playing. The crossword button is also centre stage which helps particularly younger people get right into the action and not get tilted navigating complex menus

CROSSWORD GENERATOR: Static Label acting as title text
Settings: Button. Upon press navigates to Settings Page
Crossword Select: Button. Upon press navigates to Difficulty Select page

Records: Button. Upon press navigates to Records page

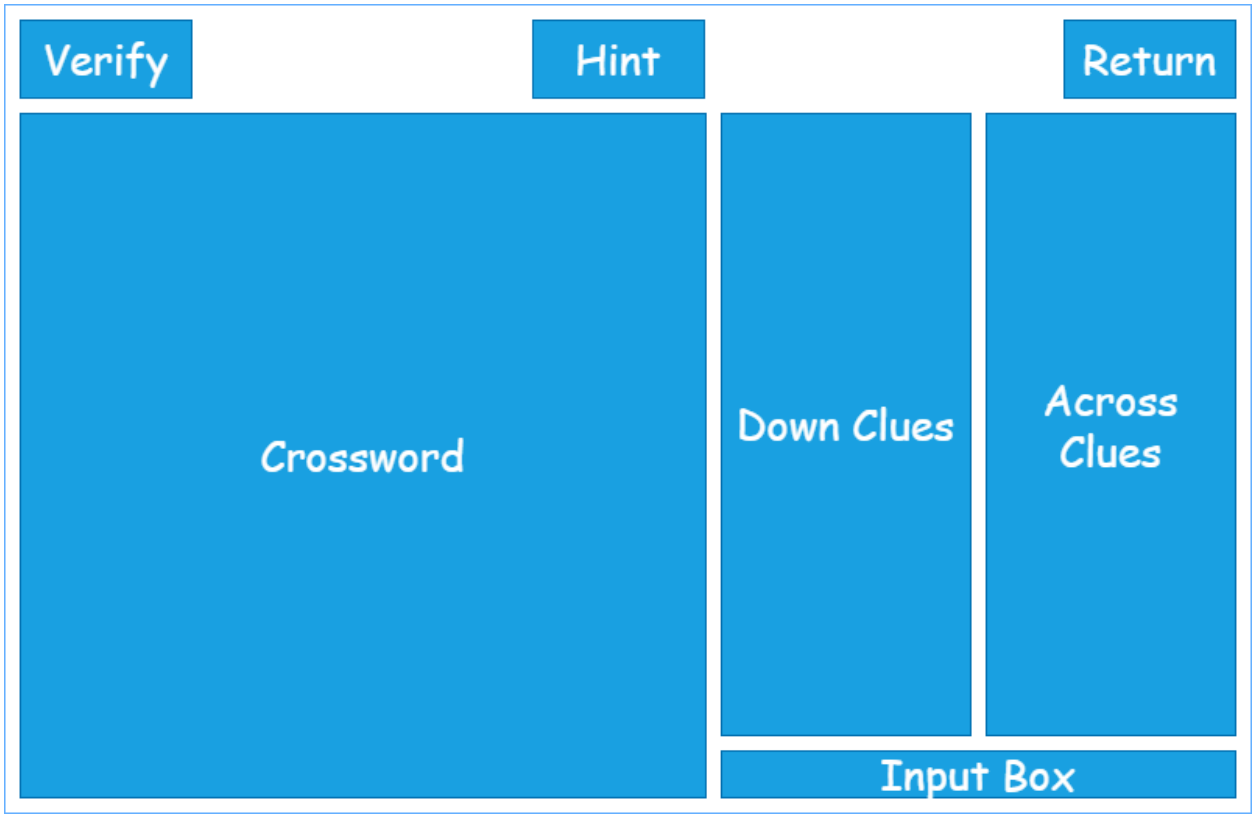
Links to success criteria:

- Easy to navigate
- Simplistic, accessibility prioritised



This is the crossword selection menu, navigated to from the homepage. The difficulties are laid out in an orderly fashion alongside competitive features bar along the bottom. After completing enough hard crosswords, CHIMPS will be unlocked. Similar to the first page more info is displayed by hovering over the buttons for long enough

Return: Button. Upon press navigates to Homepage
Difficulty Select: Static label
Easy: Button. Upon press sets difficulty to Easy and navigates to Crossword Screen
Medium: Button. Upon press sets difficulty to Medium and navigates to Crossword Screen
Hard: Button. Upon press sets difficulty to Hard and navigates to Crossword Screen
CHIMPS: Button. Upon press sets difficulty to CHIMPS and navigates to Crossword Screen



This is the Crossword Screen displayed after crossword generation is complete. It succeeds in fitting all necessary features onto one page so users don't have to go back and forth between pages. It is also considerate of smaller screens by having the words and definitions locked into a minimum font size. It is loosely based on The Guardian's design as I feel the simplistic placements work really well and it is easy to understand where things are. Just like difficulty selection I kept the return button in the top right corner to maintain consistency.

Return: Button. Upon press navigates to Homepage
Verify: Button. Upon press runs the function to check if the user's answer is correct. If correct, navigates to Victory Screen
Hint: Button. Upon press runs the function to give the user a hint to help them solve the crossword.
Crossword: Dynamic label. Displays the crossword grid and gets updated when user inputs their answer into Text Boxes. Crossword cells also highlight when user is selected on certain words.
Down Clues: List of buttons. Each displays down position ie "down 5" and definition of that word. Upon press the selected word is highlighted on the Crossword and the user is prompted to input their answer in the Input Box
Across Clues: Acts the same as Down Clues but for across answers
Input Box: Input Box. Once a word is selected from Down/Across Clues allows user input. When enter pressed, the word is displayed on Crossword if in the correct format and saved so that it can be used in the verification algorithm.

This is the settings menu. The options are separated according to what kind of setting they are and will be clearly labelled.

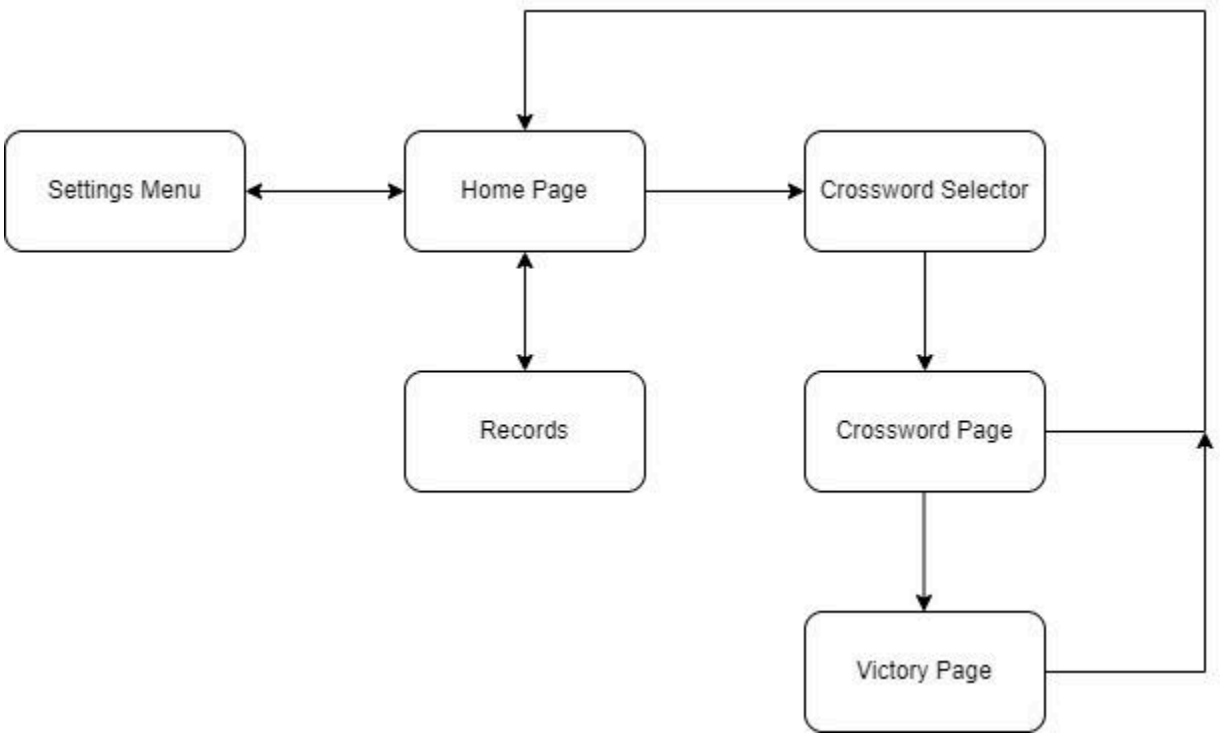
Accessibility Settings: List of toggleable buttons like font size, font type, colourblind mode
Skins/Colour Changes: List of buttons. Upon press change skin to that of the button pressed
Misc Settings: List of toggleable buttons like
Return: Button. Upon press navigates to Homepage

This is the Records page. It stays in line with being simplistic while still providing the necessary information. Cumulative records are stored on the left and Individual records are stored on the right. There is also a reset data button which does as so.

Return: Button. Upon press navigates to Homepage
Reset data: Button. Upon press asks for confirmation and if confirmed erases all user data and replaces it with a fresh slate.
Cumulative Records: A list of labels for records such as total time taken, total words answered, total crosswords completed
Individual Records: A list of labels for records such as fastest completion of each difficulty and average time for each difficulty

This is the victory page. It fits the general aesthetic and positional placements of the other windows.

Victory: Label
Time taken: Label displaying the time taken for the user to complete the crossword
Words answered: Label displaying the number of words answered
Letters answered: Label displaying the number of letters answered
Return: Button. Upon press navigates to Homepage



The figure above is a page navigation chart of my solution. The main emphasis is everything being centred around the homepage so that everything is within arms reach for the user. Throughout designing all of these diagrams abstraction was applied in the action of ignoring style choices and only focusing on the key features of each page. When developing these interfaces I will keep the same principle from the beginning and only after the main layout is complete will the page be stylised.

Usability features

The usability features in my solution are something I focused on due to the demographics I am targeting being less used to digital systems. The buttons are made large and are labelled appropriately to make it easy to see them and click on them. This is evident in the abstracted diagrams and will remain present through later iterations.

Font size is set to large by default which is particularly important in a medium which is essentially based on reading questions and answering them. The font itself will be set to comic sans by default to further ease the reading process for dyslexic people. Both of these can be toggled to more ‘normal’ settings in the main menu which is highlighted by a large button.

There will also be colourblind and high contrast modes based on online standards. It might not immediately be apparent why this is necessary because crosswords are normally in black and white however in my solution there will be much more vibrance and colour and in order for users to experience this to the highest standards modes like this are necessary.

Stakeholder Input

After listing these features out and evaluating solutions I sent my stakeholders emails to gather their feedback on the recent changes. Their responses are as follows:

Student A:

Looks pretty decent. Because CHIMPS needs an amount of hard crosswords completed first I think you should add a stat tracker in the menu somewhere. Other than that the menu looks clean and well thought out.

Student B:

It looks good so far. The UI looks clean and easy from the diagrams, easy to navigate. I expect big things in the future.

Response:

I spoke to Student A about the stat tracking idea further and we both agreed that it should record crosswords completed, words answered, letters typed, average time taken and cumulative time for each difficulty mode. To implement this I intend to reuse the CHIMPS unlock parameters and tweak them for different variables.

Student B approved of the GUI and its simple design, which is good considering ease of access was something I prioritised when making those diagrams.

Algorithms

The main bulk of this solution is the backend crossword generator itself. This can be broken down mainly into putting the words and their definitions into a database, linking letters in them together and producing this as a crossword grid. The first of these is a one time setup whereas the rest must be run every time a crossword needs to be generated. Breaking these down into even smaller subroutines we get the following:

Gather words and definitions:

- Gather dictionaries of various words and their definitions into a table
- Segment these based on crossword difficulty/mode etc

Select a list of these words based on difficulty and link them together:

- randomly select words and their definitions from selected difficulty's word pool
- From these words, analyse their letters and link them together

Produce the crossword grid:

- Produce a crossword grid from the data above
- Display the words' definitions and their respective numbers beside the crossword
- Implement a checking feature

Gathering words and their definitions:

Gather dictionaries of various words and their definitions:

Words and dictionaries must be gathered from trusted online sources.

Segment words based on difficulty:

Using a csv file the words can be divided and given tags based on what their maximum difficulty value is. This means that more advanced words with a value of 5 will not appear in the beginner crossword and to avoid oversaturation the words with a value of 4 can appear in the level 5 crossword.

Select a list of these words based on difficulty and link them together:
randomly select words and their definitions from selected difficulty's word pool:
From the csv file words will be selected based on their difficulty value. Words with the same value as the difficulty have a higher pick rate than those of a lower or higher value.

From these words, analyse their letters and link them together:
Based on the word selected, join together their letter, making sure every word has at least one joined letter in it. Validation is very important here as having a letter joined to 2 other letters can cause the crossword to be rendered unreadable

Produce the crossword grid:
Produce a crossword grid from the data above:
The data from above must be placed into a grid and fitted onto the screen. An algorithm will deduce which square to use as the central one. The First letter of each word on the grid will be assigned a number for the definition to have. Where there is overlap of the first letters the number will be shared due to one word having 2:down and the other having 2:across for example.

Display the words' definitions and their respective numbers beside the crossword:
On the right of the main crossword will be the words' definitions and numbers. Here the user can type in the words for each definition and it will show up on the grid. While selected on a word that word's space on the grid will highlight, in order to make it easier to visualise what is happening but also as a design choice.

Implement a checking feature:
If the user has inputted all of the words the program will clock it and produce victory screen

Subroutines

Purpose: To sort words into difficulties

```
FUNCTION difficultySort():
        FOR i IN wordsRaw:
                IF i IN wordsSorted == FALSE:
                        wordsSorted.append i
                        i.value = 1
                IF local(i.value) > wordsSorted(i.value):
```

```
                wordsSorted(i.value) = local(i.value)
        FUNCTION END()
```

Purpose: Select words based on difficulty

```
FUNCTION difficultyChoose(difficulty):
        FOR i IN RANGE(0, (difficulty*5 +20)):
                IF RNG(0,5) >1:
                        RNG(0,LEN(wordsSorted(n.value = difficulty)))
                        crosswordWords.append(wordsSorted(n.value = difficulty))
                ELSE:
                        RNG(0,LEN(words Sorted))
                        crosswordWords.append(wordsSorted(n))
        FUNCTION END()
```

Purpose: Analyse letters and link together

```
FUNCTION linkLetters():
        addedWords = []
        FOR i IN crosswordWords:
                word = crosswordWords(i)
                IF i == even:
                        word.direction = across
                IF i == odd:
                        word.direction = down
                IF i == 0:
                        addedWords.append(word)
                ELSE:
                        SEARCH addedWords for attribute != word.direction AS compare:
                                FOR j IN LEN(compare) and FOR k IN LEN(word):
                                        IF compare[j] == word[k] and
compare[j],compare[j-1],compare[j+1].joined == FALSE

                                                word[k].joined = compare[j] AND compare[j].joined
= word[k]
                                                addedWords.append(word)
                                        END
                                END
                        END
        RETURN addedWords
```

This works by utilising the attributes within multiple arrays to determine which direction the word is going and which letters are linked. It assigns links based on order priority currently which will tend to centralise the crossword around the first word chosen. Slight validation is done in order

to ensure that no neighbouring letters can be individually linked but more will be done later in cases of words overlapping at their ends.

Purpose: Verify if answer correct

```
FUNCTION verify():
        FOR i IN LEN(crosswordWords):
                IF crosswordWords(i) != userAnswer(i):
                        RETURN FALSE
        RETURN TRUE
        FUNCTION END()
```

Check the crossword for any incorrect matches. If none are found the function is allowed to carry out and return TRUE.

Purpose: Give user hint

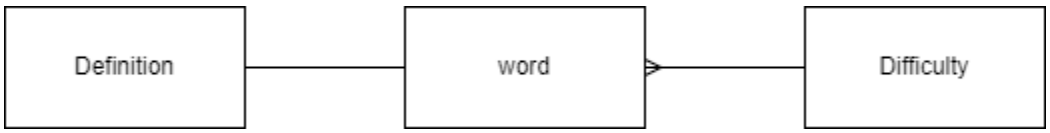
```
FUNCTION hint():
        error = FALSE
        WHILE error == FALSE:
                i = RNG(0,LEN(crosswordWords))
                IF crosswordWords(i) != userAnswer(i):
                        j = RNG(0, LEN(crosswordWords(i)))
                        INSERT(crosswordWords(i)(j)) ONTO BOARD
                        error = TRUE
        FUNCTION END()
```

This works by randomly selecting a word out of the number of words and then from this word randomly selecting a letter. This letter is then placed onto the grid.

Permanent Data Storage

My solution requires a large volume of data to be stored which must be retrieved every time the user wishes to generate a new crossword. Based on this criteria a database would be best suited to holding the permanent data.

Each word stored has its own unique definition and also has one difficulty shared with other words. The entity-relationship diagram is shown below



Key Variables

Data item	**Data type**	**Explanation**
wordsRaw	Record	Unsorted compilation of all words gathered. Stores word (string), definition (string) and difficulty (integer).
wordsSorted	Record	wordsRaw but sorted and normalised. Stores the same fields but unlike wordsRaw data from this will be retrieved during runtime.
crosswordWords	Array of dictionaries	Used to store all of the words that will be used on the grid during runtime
word	Dictionary	Stores word (string), direction (string) and joined letters (int)
userAnswers	Array of strings	Stores the user's inputs compares these against crosswordWords to determine if completed
Records	Record	Stores the number of completions (int), correct words (int), correct letters (int), average time taken (int), cumulative time taken (int) for each difficulty. Note that the latter 2 will need conversions for display
selectedDifficulty	int	The user selects this and the generated crossword is

		influenced by its value

Data item	Data type	Explanation
wordsRaw	Record	Unsorted compilation of all words gathered. Stores word (string), definition (string) and difficulty (integer) stored in a file
wordsSorted	Record	Stores the same fields as wordsRaw but unlike the latter is sorted and has no unnecessary duplicates
crossword	2d array	A 30 by 30 grid of the crossword. Unused spaces are occupied by “*”. Generated in the letterLink function
directionsacross	2d array	The position, length and definition of each word. Is used to create the clues and the positional data is used to verify/highlight the grid. Only across words
directionsdown	2d array	directionsacross but for downwards words
lastSelected	array	The last selected word on the grid from either directionsdown or directionsacross. Used to unhighlight words in the highlight function

palette	array	The current selected colour values. Change during the colour_change function and are then used to update widget colours in colour_update

Validation

In order to make sure the program functions, all user inputs must be validated so that they do not break anything.

Buttons

The buttons I use will be … The user will not be able to input any custom text through any of these so minimal validation is needed.

Answer box

These accept any user input and as such will need the most validation done. In the context of the answer box only alphabetical characters should be accepted. There should also be a limit on the length of input depending on the number of tiles that word takes up on the grid in order to help the user if they miscount. A presence check is unnecessary for this field because the user should be allowed to clear one of their answers if they wish. The algorithm will be adjusted to ensure the program doesn't crash when the field is left blank.

Settings

Testing work must be done with the settings, particularly when more than one accessibility option is toggled at once. This is done to ease the user experience.

Testing method

Each function of the project should be tested thoroughly both individually and as a whole to show that the system objectives are met.

During development tests should be run to ensure input data provides the intended output data. Any unexpected results or crashes should be screenshotted alongside the inputs and analysed to determine the cause.

After development final test shall be done alongside these to ensure no new errors have cropped up as new content was added

Testing will be done with normal data to ensure the program works properly when intended data is received. The same will be done with erroneous data which is data intended to be rejected. This is for the purpose of making sure no illegal data gets through the system. Finally boundary inputs will be tested to make sure no errors slide through the cracks. This is arguably most important as it could potentially render some necessary inputs uninputable which could potentially cause an unsolvable crossword for example.

Test plan

Crossword screen answer box:

Test ID	Description	Test data	Intended result
01	Check that user can input alphabetical words of correct length	On answer field of length 6 Answer = ‘monkey’ Press enter	‘monkey’ appears in field
02	Check that user can’t input alphabetical words shorter than intended	On answer field of length 4 Answer = ‘ape’ Press enter	Program refuses input
03	Check that user can’t input alphabetical words longer than intended	On answer field of length 7 Answer = ‘bigchungus’ Press enter	Program refuses input
04	Check that user can’t input purely numerical characters	On answer field Answer = ‘777’ Press enter	Program refuses input
05	Check that user can’t input numerical and alphabetical hybrid string	On answer field Answer = ‘g0rill4’ Press enter	Program refuses input

Verification and hint Button:

Test ID	Description	Test data	Intended result
06	Check that the verification button accepts a completed crossword	All answer fields correctly filled in Press verification button	Victory screen displayed
07	Check that the verification button does not accept an incorrect crossword	All answer fields filled in: one mistake in crossword Press verification button	Displays crossword incorrect
08	Check that verification button does not work if empty space in crossword	Some answer fields filled in, not all. Press verification button	Displays that there are spaces left to be filled in
09	Check that hint button fills in a character if pressed in non-extreme difficulty	In non-extreme mode Answer fields semi-complete or incomplete Press hint button	A random character is filled in and highlighted blue

Difficulty:

Test ID	Description	Test data	Intended result
10	Check that easy can be inputted and functions	Select easy Press play	Value 1 words appear
11	Check that extreme can be selected and functions	Select extreme Press play	Value 4 words appear

Stat tracker:

Test ID	Description	Test data	Intended result
12	Check that letters answered, words answered and the difficulty of the crossword completed are correctly recorded	On a fresh save Select difficulty Press play Record how many words and letters are answered. Correctly verify	The total words and total letters in the file should be equal to what was recorded. The difficulty completed should have 1 completion
13	Check that average and cumulative time taken is correctly recorded	On a fresh save Select difficulty, press play Complete crossword in 5 minutes Complete a second crossword of the same difficulty in 9 minutes Go to records	Selected difficulty should have ~7 minutes under average time taken and ~14 minutes under cumulative time taken. Stored as an integer
14	Check that the first time recorded is always added to the file	Reset data Complete a crossword in any amount of time	The difficulty completed's fastest value is that of the run you just did
15	Check that a faster timer replaces a slower time on a difficulty	Set the fastest easy difficulty to 20 minutes Complete an easy crossword in less than 20 minutes	The new time recorded is displayed as fastest in Records Page. Stored as an integer
16	Check that a slower time does not replace a faster time on a difficulty	Set the fastest medium difficulty to 2 minutes Complete a medium crossword in more than 2	The fastest time in Records Page is still 2 minutes

Records Page:

Test ID	Description	Test data	Intended result

17	Check that if no time is recorded the Records Page displays it correctly	Reset data View records	The record page displays any uncompleted crosswords's times as "N/A". Everything else is zero
18	General records (not specific to certain difficulties) are displayed correctly	Fill every value in scores.csv View records	Values are displayed next to their names. Any time values are displayed in hh:mm:ss format.
19	Difficulty specific records are displayed correctly	Fill every value in scores.csv View records	Values are displayed next to their names. Fastest times are in hh:mm:ss format
20	Time based records are displayed correctly in extreme cases	Set scores.csv time's to various high values. Include those less than and more than 3600 to ensure hours are tested	Values are displayed correctly in hh:mm:ss

Settings Page:

Test ID	Description	Test data	Intended result
21	Check that pressing a skin button changes the skin	Press dark mode button	Widgets and background are changed to that of dark mode
22	Check that high contrast mode works	Press high contrast mode	Widgets and background are changed to that of high contrast mode

23	Check that you cannot equip multiple skins at once	Press dark mode button Press high contrast button	Widgets and background are changed to that of dark mode
24	Check that colourblind modes work	Press a colourblind button	Widgets and background are changed to that of selected colourblind mode

25	Upon clue selection that clue is highlighted on the grid	Select crossword Select clue Select another clue	Correct clue is highlighted on the grid. When changed, reverts the old clue and highlights the new one
28	Check that tutorial can be opened, displays correctly and can be closed	Press tutorial button Press exit tutorial button	Tutorial opens and displays correctly. Upon exit press closes and leaves user where they pressed the button

UI:

Test ID	Description	Test data	Intended result
26	Check that victory page displays the correct data from the just completed crossword	Enter a crossword of any difficulty. Count the number of words and letters and record time taken. Verify crossword	Words and letters answered are the same as what was on the crossword. Time taken is also the same and is validated into hours, minutes and seconds.
27	Check that navigational	Press select difficulty	Select difficulty goes to

	buttons on the homepage function	return Press settings return Press records	difficulty page Settings goes to settings page Records goes to records page
28	Check that font changing button works	Press font change button Press font change button	Switches to the other font on first press. Switches back upon second

If all of these are working as intended I will know that my program is functional and complete

Once all of the tests are completed and confirmed working I will know that the program is sufficiently validated to correct standard.

Development and Testing

Overview

Due to the nature of my project and how interlinked each function is I decided to develop each function and procedure individually. Abstraction was applied on top of this, simplifying the problem into feeding in a set of inputs and producing a set of outputs. This was done to make it much easier to individually develop each section and also makes it easier to plug in data to test without running the entire project.

During development there were times where the determined inputs and outputs for each function would need to change and it is noted where that has occured

Iteration 1:

The main objective of this iteration is to have the individual components all coded and working but not necessarily validated. This includes having a GUI which can be navigated through but not yet with the entire functionality of the programs: they will be implemented later.

sortDirections

```
words = ["monkey", "john","ape", "jack","drilla"]

def assignDirection(words):
    directions = {}
    for i in range(0,len(words)):
        word = words[i]
        if i % 2 == 0:
            directions[word] = True   #True means horizontal, False means vertical
        else:
            directions[word] = False
    return directions
```

This simple algorithm determines the direction of each word based on index of the list. The main use of this is assigning them into a dictionary so that the program can easily search by key (direction) during the linking letters stage.

```
{'monkey': True, 'john': False, 'ape': True, 'jack': False, 'drilla': True}
```

This is a screenshot of what the dictionary currently looks like.

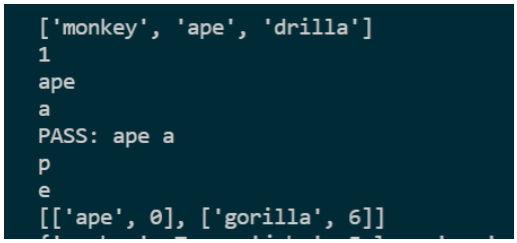
linkLetters:

```
def letterSort(directions):
    passed = False
    word = "gorilla"
    addedWords = [""]
    toCompare = []
    for i,j in directions.items():
        if j == True:
            toCompare.append(i)
    print(toCompare)
    while passed == False:
        k = r.randint(0,len(toCompare)-1)
        print(k)
        print(toCompare[k])
        for l in range(0,len(toCompare[k])):
            print(toCompare[k][l])
            for m in range(0,len(word)):
                if passed == False:
                    if word[m] == toCompare[k][l]:
                        print("PASS:",toCompare[k], word[m])
                        passed = True
                        link1 = [toCompare[k],l]
                        link2 = [word,m]
        toCompare.remove(toCompare[k])
    return(link1,link2)
```

This is the initial algorithm to pseudorandomly link the letters. It utilises the directions dictionary created earlier to make sure to not match 2 words of the same direction by creating a new list (toCompare).

It uses the random function to select a random word from toComapre to be tested against. This word's letters are compared against that of the word we are trying to add to the list. If the letters match 2 arrays are created, one for the new word and one for the old one. The loop is then ended and they are passed out of the function. If no match is found amongst a word that word is temporarily removed from the comparison list. Although not implemented at this stage there will be more failsafing done in case there is no match amongst any of the words.

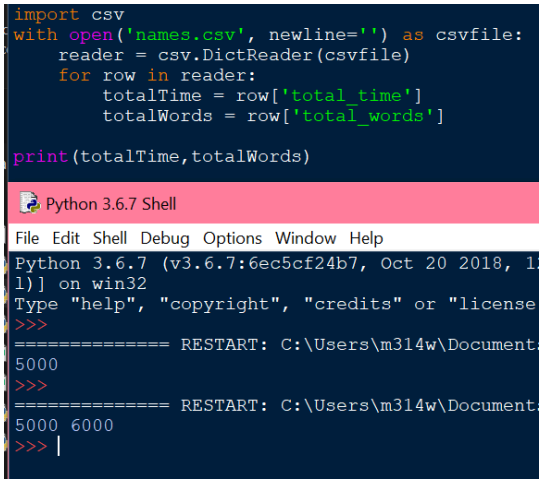
Currently I've decided to only randomise the word testing order since in the final project there will be so many different words being compared that any more randomness will have diminishing returns in its value. Sorting in order may also make the validation algorithms easier to implement if they.



STATTRACK:

```python
import csv

def statTrackGather(crosswordWords):        #run once crossword completed
    timeTaken = end-start
    wordsCompleted = len(crosswordWords)
    lettersCompleted = 0
    for i in range(0,wordsCompleted):
        lettersCompleted +=crosswordWords[i]

def statTrackAppend(timeTaken,wordsCompleted,lettersCompleted):
    with open('scores.csv', newline='') as csv:
        reader = csv.DictReader(csvfile)
        for row in reader:
            totalTime = row['total_time']
            totalWords = row['total_words']
            letterTotal = row['total_letters']

    averageTime = (averageTime+timeTaken)/2
    totalTime += timeTaken
    wordTotal += wordsCompleted
    letterTotal += lettersCompleted
```

Here is the first algorithm designed for the stat tracking feature. At this stage I've decided on using a csv file to store all user data due to efficient and easy lookup of items based on key. Currently this algorithm assigns the correct variables and the future iteration will involve overwriting the old ones with the new ones in the file.



FIRST TIME SETUP:

In order to add to a file the file must exist in the first place. This is needed for the stat tracker and I determined a function which would run if the scores.csv doesnt exist to be the best method.

```
import csv
def firstTimeSetup():   #Setup stattrack values, unlocks, etc
    with open('scores.csv','w' ,newline='') as csvfile:
        fieldnames = ['total_time','average_time','total_words',
        writer = csv.DictWriter(csvfile,fieldnames=fieldnames)



        writer.writerow({'total_time':'total_time','average_time
        writer.writerow({'total_time':'0','average_time':'null',

        csvfile.close()


try:
    with open('scores.csv','r',newline='') as csvfile:
        pass
except:
    firstTimeSetup()
'
```

The procedure firstTimeSetup adds the necessary headers and default data to the file. A try, except statement is used to determine whether to run the procedure or not by checking if it already exists.

LETTERLINKS:

This improved algorithm uses a grid and essentially does both parts at once.

The biggest errors I encountered were to do with the indices of the split lists which I added to the front and back of the letter on the grid. Initially I attempted to search by reverse index with a for loop ie

FOR i IN RANGE(LEN(word)):
    OUTPUT WORD[-i]

I switched my method to using ::-1 to reverse the string instead which made it both functional and more easy to read.

There were also many errors due to which direction the words were going, particularly with the downwards ones.

```
* : m
o : o
* : n
x : k
* : e
* : y
```

Output screenshot of the program comparing grid spaces. If they are either the same character or the board is blank, marked by the asterisk, the program will add the new character to the board. Outer regions are searched too but they were initially functional so not listed in the screenshot. In the case above the program should not return true because in the 4th line x != k

```
for j in range(len(post)):
    print(grid[position[0]+j][position[1]]," : ",post[j])
    newgrid[position[0]+j][position[1]] = post[j]
    if (grid[position[0]+j][position[1]] == "*" or grid[position[0]+j][position[1]] == post[j]) == False or grid[position[0]+j][position[1]-1] != "*" or grid[position[0]+j][position[1]+1] != "*":
        horizontal = False
        print("false",j)
```

```
for j in range(len(post)):
    print(grid[position[0]+j][position[1]]," : ",post[j])
    newgrid[position[0]+j][position[1]] = post[j]
    if (grid[position[0]+j][position[1]] == "*" or grid[pc
        horizontal = False
        print("false",j)
```

I also had the error of trying to add the new letters to the grid before searching which resulted in a bunch of incorrect readings until I fixed this and moved it to add letters after searching.

```
[['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'g', 'o', 'r', 'r', 'i', 'l', 'l', 'a', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*']]
```

^initial grid

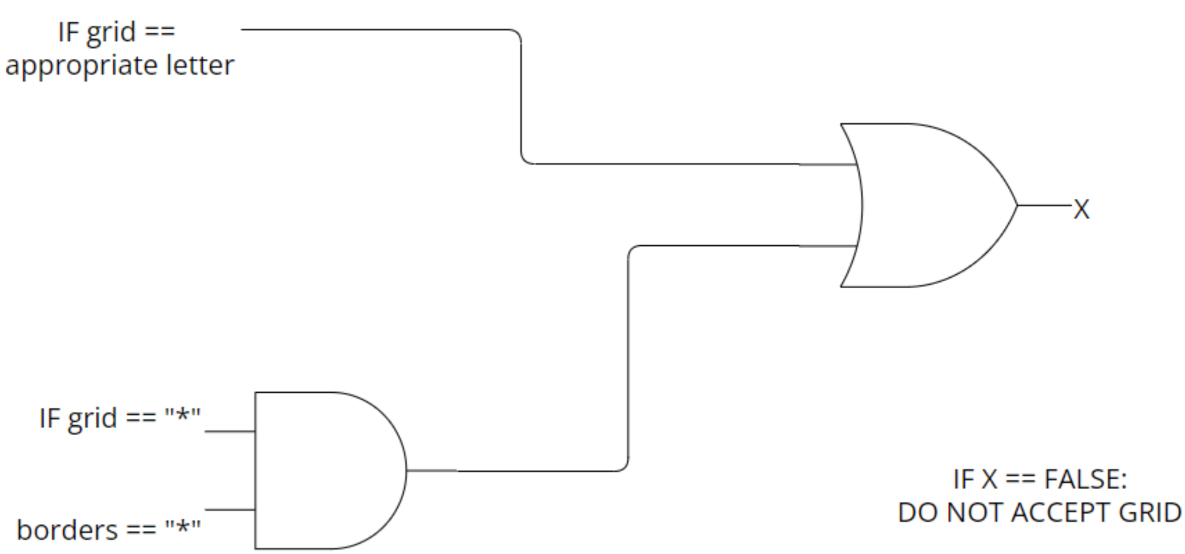
```
[['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', 'm', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'g', 'o', 'r', 'r', 'i', 'l', 'l', 'a', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', 'n', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', 'k', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', 'e', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', 'y', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*']]
```

^^intended result but passed as a fail

There was another issue I encountered when trying to connect whole words I didn't encounter while using single letters on the grid and that was the actual letter the program was trying to link on the board (in this case "o") isn't treated any differently to the more secluded letters, meaning the program sees the "g" and "r" next to the "o" and marks the linking as a fail as it only accepts "*" as a bordering character.

Having this error also made me realise that the same issue could occur when a word has multiple links in it and a fix can't just be exclusive to the first index of the "post" substring but must fit all letter overlaps on the board.

I produced a logic diagram to easier visualise what was going on with my program:



I applied this to my program and ended up with this correctly working statement, solving the problem.

```
if ((grid[position[0]-i-1][position[1]] != "*" and (grid[position[0]-i-1][position[1]+1] != "*" or grid[position[0]-i-1][position[1]-1] != "*")) and grid[position[0]-i-1][position[1]] != pre[i]):
    horizontal = False
    print("false",i)
```

Letters neighbouring the start and end of words:

If letters from different words neighbour each other at the start and end it gives the impression of them being one long word. My program wants to avoid this as it ruins the visibility of the crossword therefore harming the user experience.

I figured the easiest way to implement this would be to add an or statement for the grid positions of the main letter plus or minus the length of pre and post respectively.

```
[['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'g', 'o', 'r', 'r', 'i', 'l', 'l', 'a', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', 'x', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*']]
```

This solution was tested on the above grid which if executed correctly should never return true.

```
TESTING:
*  :  m
o  :  o
*  :  n
*  :  k
*  :  e
*  :  y
ends hit
pre: *
post: x
[['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', 'm', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'g', 'o', 'r', 'r', 'i', 'l', 'l', 'a', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', 'n', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', 'k', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', 'e', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', 'y', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', 'x', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*']]
```

^^ This is the output produced by my program with description included for easier testing. By following the validation done in the above section it is clear that the initial tests do not flag and are therefore successful. The program successfully flags that one of the ends has a collision and upon observing the feedback we can see the post or end of the word collides. Because this successfully detects this and returns false I can deem this algorithm working and must implement it to the horizontal function as well.

```python
if grid[position[0]-len(pre)-1][position[1]] != "*" or grid[position[0]+len(post)][position[1]] != "*":
    horizontal = False
    print("ends hit")
print("pre:", grid[position[0]-len(pre)-1][position[1]])
print("post:", grid[position[0]+len(post)][position[1]])
```

Also to be noted the horizontal and vertical functions differ in which index is searched. They're essentially reversed and the algorithm is almost identical between them. An example is

```python
for i in range(len(pre)):
    print(grid[position[0]-i-1][position[1]], " : ",pre[i])
```

becomes

```python
for i in range(len(pre)):
    print(grid[position[0]][position[1]-i-1], " : ",pre[i])
```

The same applies for the bordering squares which have a value of +-1.

TESTING DIRECTIONS:

There is no need to test erroneous data here because all “inputs” will be fed by the validated databases.

Tests to run:

Test ID	**Description**	**Expected outcome**
1	Check that program passes true when linking a single word with minimum one similar character	Returns True
2	Check that program passes true when crossing through another word with same character in position	Returns True
3	Check that program passes false when going through a different character	Returns False
4	Check that program passes false when bordering a different character and not passing through another word	Returns False
5	Check that program passes false when there is a character bordering the start or end of the word	Returns False

Vertical 1:

Original grid

```
[['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'm', 'o', 'n', 'k', 'e', 'y', 'l', 'a', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*']]
onkey
```

Output grid

```
PAASSEED
[['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', 'm', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'g', 'o', 'r', 'r', 'i', 'l', 'l', 'a', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', 'n', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', 'k', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', 'e', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', 'y', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*']]
```

Successfully passed with no issues

Vertical 2:

Original grid

```
[['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'm', 'o', 'n', 'k', 'e', 'y', 'l', 'a', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', 'k', 'e', 'y', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*']]
```

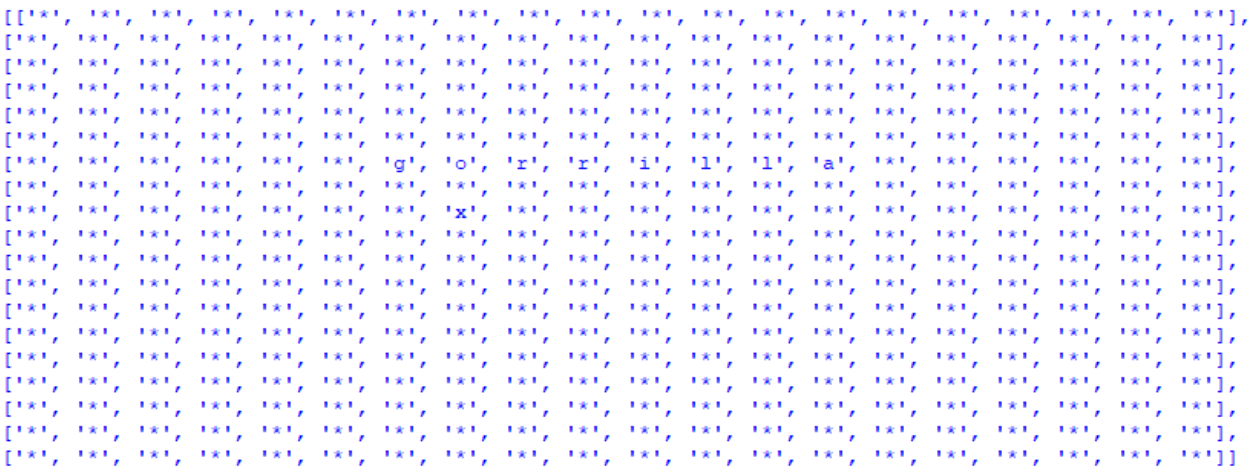
Output grid

```
PAASSEED
[['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', 'm', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'g', 'o', 'r', 'r', 'i', 'l', 'l', 'a', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', 'n', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', 'k', 'e', 'y', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', 'e', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', 'y', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*']]
```

Successfully passed with no issues

Vertical 3:

Original grid



Output grid

Successfully passes as False and flags the correct character as incorrect

Vertical 4:

Original grid

```
[['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'm', 'o', 'n', 'k', 'e', 'y', 'l', 'a', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', 'x', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*']]
```

Output

```
TESTING:
*  :  m
o  :  o
*  :  n
*  :  k
false 2
*  :  e
*  :  y
pre: *
post: *
[['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*']
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', 'm', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'g', 'o', 'r', 'r', 'i', 'l', 'l', 'a', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', 'n', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', 'k', 'x', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', 'e', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', 'y', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*']]
```

Successfully passes as False and flags the correct character as incorrect

Vertical 5.1:

Original grid

```
[['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'g', 'o', 'r', 'r', 'i', 'l', 'l', 'a', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', 'x', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*']]
```

Output grid

```
TESTING:
*  :  m
o  :  o
*  :  n
*  :  k
*  :  e
*  :  y
ends hit
pre: *
post: x
[['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', 'm', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'g', 'o', 'r', 'r', 'i', 'l', 'l', 'a', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', 'n', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', 'k', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', 'e', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', 'y', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', 'x', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*']]
```

Successfully flags as false and displays it as post.

Vertical 5.2:

Original grid

```
[['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', 'x', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'g', 'o', 'r', 'r', 'i', 'l', 'l', 'a', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*']]
```

Output

```
TESTING:
*  :  m
o  :  o
*  :  n
*  :  k
*  :  e
*  :  y
ends hit
pre: x
post: *
[['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', 'x', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', 'm', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'g', 'o', 'r', 'r', 'i', 'l', 'l', 'a', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', 'n', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', 'k', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', 'e', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', 'y', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*']]
```

Successfully flags as false and displays it as pre.

Horizontal 1:

Original grid

```
[['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'g', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'o', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'r', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'r', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'i', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'l', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'l', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'a', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*']]
```

Output

```
[['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'g', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', 'm', 'o', 'n', 'k', 'e', 'y', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'r', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'r', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'i', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'l', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'l', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'a', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*']]
PAASSEED
```

Successfully passes the correct grid

Horizontal 2:

Original grid

```
[['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'g', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'o', '*', 'k', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'r', '*', 'o', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'r', '*', 'o', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'i', '*', 'l', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'l', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'l', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'a', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*']]
```

Output

```
[['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*']
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'g', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', 'm', 'o', 'n', 'k', 'e', 'y', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'r', '*', 'o', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'r', '*', 'o', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'i', '*', 'l', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'l', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'l', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'a', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*']]
PAASSEED
```

Successfully passes as intended

Horizontal 3:

Original grid

```
[['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'g', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'o', '*', 'x', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'r', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'r', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'i', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'l', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'l', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'a', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*']]
```

Output

```
TESTING:
*  :  m
o  :  o
*  :  n
x  :  k
false 2
*  :  e
*  :  y
pre: *
post: *
DONE
=====================================================
[['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'g', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', 'm', 'o', 'n', 'k', 'e', 'y', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'r', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'r', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'i', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'l', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'l', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'a', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*']]
```

Successfully passes as fail and notes due to k = x at the correct spot

Horizontal 4:

Original grid

```
[['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'g', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'o', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'r', '*', 'x', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'r', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'i', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'l', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'l', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'a', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*']]
```

Output

```
TESTING:
*  :  m
o  :  o
*  :  n
*  :  k
false 2
*  :  e
*  :  y
pre: *
post: *
DONE
=====================================================
[['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'g', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', 'm', 'o', 'n', 'k', 'e', 'y', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'r', '*', 'x', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'r', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'i', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'l', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'l', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'a', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*']]
```

Successfully passes as fail and notes it due to an error at “k” correctly

Horizontal 5.1:

Original grid

```
[['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*']
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'm', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'o', '*', '*', '*', '*', 'x', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'n', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'k', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'e', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'y', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'l', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'a', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*']]
```

Output

```
TESTING:
*  :  m
o  :  o
*  :  n
*  :  k
*  :  e
*  :  y
ends hit
pre: *
post: x
DONE
======================================================
[['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'g', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', 'm', 'o', 'n', 'k', 'e', 'y', 'x', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'r', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'r', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'i', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'l', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'l', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'a', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*']]
```

Successfully rejects grid and notes it due to border at the end of the word

Horizontal 5.2:

Original grid:

```
[['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'g', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', 'x', '*', 'o', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'r', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'r', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'i', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'l', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'l', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'a', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*']]
```

Output:

```
TESTING:
*  :  m
o  :  o
*  :  n
*  :  k
*  :  e
*  :  y
ends hit
pre: x
post: *
DONE
=====================================================
[['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'g', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', 'x', 'm', 'o', 'n', 'k', 'e', 'y', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'r', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'r', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'i', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'l', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'l', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', 'a', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*']]
```

Successfully rejects grid and notes it due to border at the beginning of word.

Side note that the current dimensions of the grid are 20x20. This is mainly at this level to make testing easier but future grids can be easily altered at this point in time so the final number will be decided in the future. A further note is that it may be worth adding different size grids for different difficulties but this will be tested and done later on.

Initial Setup:

Now that the letter linking is functional the program needs the first word to be placed on the grid and to build on it. I decided this will work by placing the word at the centre of the grid in order to minimise the chance of words hitting the grid's edges.

```
def initialAdd(word,grid):
    for i in range(0,len(word)):
        grid[20][16+i] = word[i]

    return grid
```

This is the first iteration of it which uses a for loop to map each letter to the grid and then returns it.

Word Data:

The next addition to this function was to make it pass back data for the user input section. It passes the direction, start position and word as a string in order to assign them numbers for the grid and definitions tab. This was relatively easy to implement by reusing the code from before. The values are determined at the same time as the most recent vertical or horizontal check is done so that they are not mixed up.

```
direction = "d"
startPos = position[0],position[1]-len(pre)
print("========================================
print(oldgrid)
if checked == False:
    oldgrid = copy.deepcopy(grid)
    newgrid, checked = horizontalCheck(oldgrid,p
    direction = "a"
    startPos = position[0]-len(pre),position[1]

if checked == True:
    print("PAASSEED")
    grid = newgrid
    start = direction+str(startPos)
    return newgrid,checked,(start+word)
```

```
for i in range(0,len(words)):
    grid,checked,start = gridAdd(words[i],grid)
    directions.append(start)
```

These values are also needed from the initial setup function and were implemented as so:

```
def initialAdd(word,grid):
    for i in range(0,len(word)):
        grid[20][16+i] = word[i]
    start = "a [20,16]"+word
    return grid, start


grid,start = initialAdd("gorilla",grid)
directions = [start]
```

An example of the output looks like this:

```
['a [20,16]gorilla', 'a(19, 17)monkey', 'a(20, 22)ape', 'a(13, 16)exciting']
```

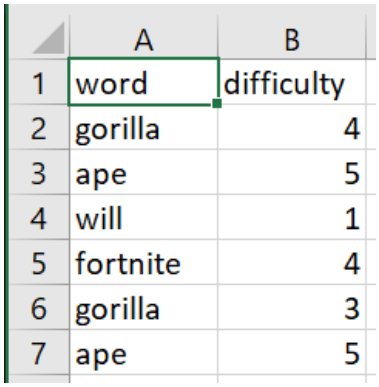
This data can be easily parsed and utilised by future function so for the time being I am content with leaving it as so.

Word Sort

Although a backend program and not provided to the user this will be necessary if the database ever corrupts or if I wish to add more words from more dictionaries. The program simply reads the word and difficulty values from the CSV file and depending on whether already searched will add them to the new file or compare the difficulties against each other. The lower difficulty takes priority and is then sent to replace the old one in the sorted dictionary.

The dictionary data type is used here for easy lookup based on key therefore making for easy comparison. The first iteration of this program looks like this

```
import csv
def gatherRaw():
    with open('wordsRaw.csv', newline='') as csvfile:
        sortedWords = {'will':5,'ape':3}
        reader = csv.DictReader(csvfile)
        fieldnames = ['word','difficulty']
        for row in reader:
            if row['word'] in sortedWords:
                if int(row['difficulty']) < sortedWords[row['word']]:
                    sortedWords[row['word']] = row['difficulty']
            else:
                sortedWords[row['word']] = row['difficulty']
    return sortedWords
sortedWords = gatherRaw()
print(sortedWords)
```



	A	B
1	word	difficulty
2	gorilla	4
3	ape	5
4	will	1
5	fortnite	4
6	gorilla	3
7	ape	5

Upon performed on this file it produces the following output:

```
{'will': '1', 'ape': 3, 'gorilla': '4', 'fortnite': '4', 'gorilla ': '3'}
```

The output did not come out as expected as gorilla which was supposed to be replaced was added twice. Upon further investigation I discovered that the cause of this was blank spaces in the CSV file and not the code itself. Extra care will have to be taken when compiling the sorted words and I will likely need to validate inside of this function to prevent any future errors and perform further tests.

First Time Setup

A thought I realised further down development is that there must be a first time setup for new users and also in the case someone deletes or corrupts one of the existing files.

The requirements of this algorithm must be to create the scores file and also the user info file and set them up accordingly. Both these files will need lookup by key so I've decided again to use CSV files in order to facilitate this.

```python
import csv
def firstTimeSetup():   #Setup stattrack values, unlocks, etc
    with open('wordsRaw.csv','w' ,newline='') as csvfile:
        fieldnames = ["monkey"]
        writer = csv.DictWriter(csvfile,fieldnames=fieldnames)


        for i in range(0,5):
            writer.writerow({'monkey':'2'})
        csvfile.close()

firstTimeSetup()
```

This initial setup was developed to make sure the file can be successfully created and added to, producing this file

	A	B
1	2	
2	2	
3	2	
4	2	
5	2	
6		

Knowing that it will work all that needs to be done is add the appropriate fields in as efficiently as possible.

```
import csv
def firstTimeSetup():   #Setup stattrack values, unlocks, etc
    with open('scores.csv','w' ,newline='') as csvfile:
        fieldnames = ['total_time','average_time','total_words','total_letters','easy_completions',
        writer = csv.DictWriter(csvfile,fieldnames=fieldnames)



        writer.writerow({'total_time':'total_time','average_time':'average_time','total_words':'tota
        writer.writerow({'total_time':'0','average_time':'null','total_words':'0','total_letters':'

        csvfile.close()

firstTimeSetup()
```

	A	B	C	D	E	F	G	H	I
1	total_time	average_t	total_wor	total_lette	easy_com	medium_c	hard_com	CHIMPS_completions	
2	0	null	0	0	0	0	0	0	
3									

Here is the 2nd iteration of this function which includes the variables for scores.csv. They are all set to zero apart from average_time. This is because it will need its own first time setup sort of function within stat track because I can't use zero to calculate the average as it'd be half of the true one. For now it is easiest to use and see it as 'null' and possibly change it when stat track is finished.

Difficulty Sort:

Before I sort the words I need to actually have the words to sort. I gathered these files as pdfs and parsed them accordingly so that I could insert them into wordsRaw.csv

```
import csv
toAdd = []
header = ['word','def']
with open("words.txt") as file:
    with open('wordsRaw.csv','w', newline='') as csvfile:
        writer = csv.writer(csvfile)
        writer.writerow(['word','definition','difficulty'])
        lines = file.readlines()
        for line in lines:
            if "âš" in line:
                x = line.split(' ')
                toAdd.append(x[1])
            if "Meaning" in line:
                y = line.split('- ')
                toAdd.append(y[1])
                toAdd.append('4')
                print(toAdd)
                writer.writerow(toAdd)
                toAdd = []
```

This was the initial converter. Future iterations won't need the headers to be written and also must append to the file, not make a brand new one like this algorithm. One might also notice the difficulty being assigned as 4. This will be handpicked for each dictionary based on its complexity and I will have my stakeholders input feedback for each one in order to not have any bias.

	A	B	C
1	word	definition	difficulty
2	Adamant	hard, inflexible	4
3	Adverse	unfavorable, hos	4
4	Accord	agreement	4
5	Affected	artificial, pretend	4
6	Allay	calm, pacify	4
7	Adulterate	make impure	4
8	Agility	nimbleness, alac	4
9	Admonish	warn, reprove	4
10	Assuage	ease, lessen	4
11	Alleviate	relieve, assuage	4
12	Altruism	unselfish devotic	4
13	Amass	collect	4
14	Acrimonious	stinging, caustic	4
15	Anomaly	irregularity	4
16	Aptitude	fitness, talent	4
17	Alimony	payment to divor	4
18	Atone	make amends	4

Select words by difficulty:

This is the algorithm which will randomly select words based on difficulty from the file. Easy, normal and hard will function so that they will only accept words of that difficulty or below. CHIMPS acts as a very difficult challenge to the user and will have a more complex algorithm to it which focuses on selecting harder words. The number of words selected is based on the difficulty. The current algorithm is (15 + difficulty * 5) words but may be changed in the future based on tests and user feedback.

The program here works by using the random function within the range of that difficulty to select a random word. The program takes the word, definition but not the difficulty as within the letter linking algorithm all words are treated equal.

The CHIMPS algorithm is more spontaneous and rolls a dice which greatly favours the more difficult words. There are also no hints in CHIMPS to further add to its prestige. This is done so that the user may have more satisfaction when beating it and doesnt detract from the accessible nature of the rest of the application.

GUI:

This was the most unfamiliar part to implement. I decided to use Kivy due to its ability to dynamically change UI features while the application is open. It also allows programs to be compiled as .exe files which is essential when catering to a more general audience.

Kivy allowed me to easily link windows with Screenmanager and pass variables through different windows.

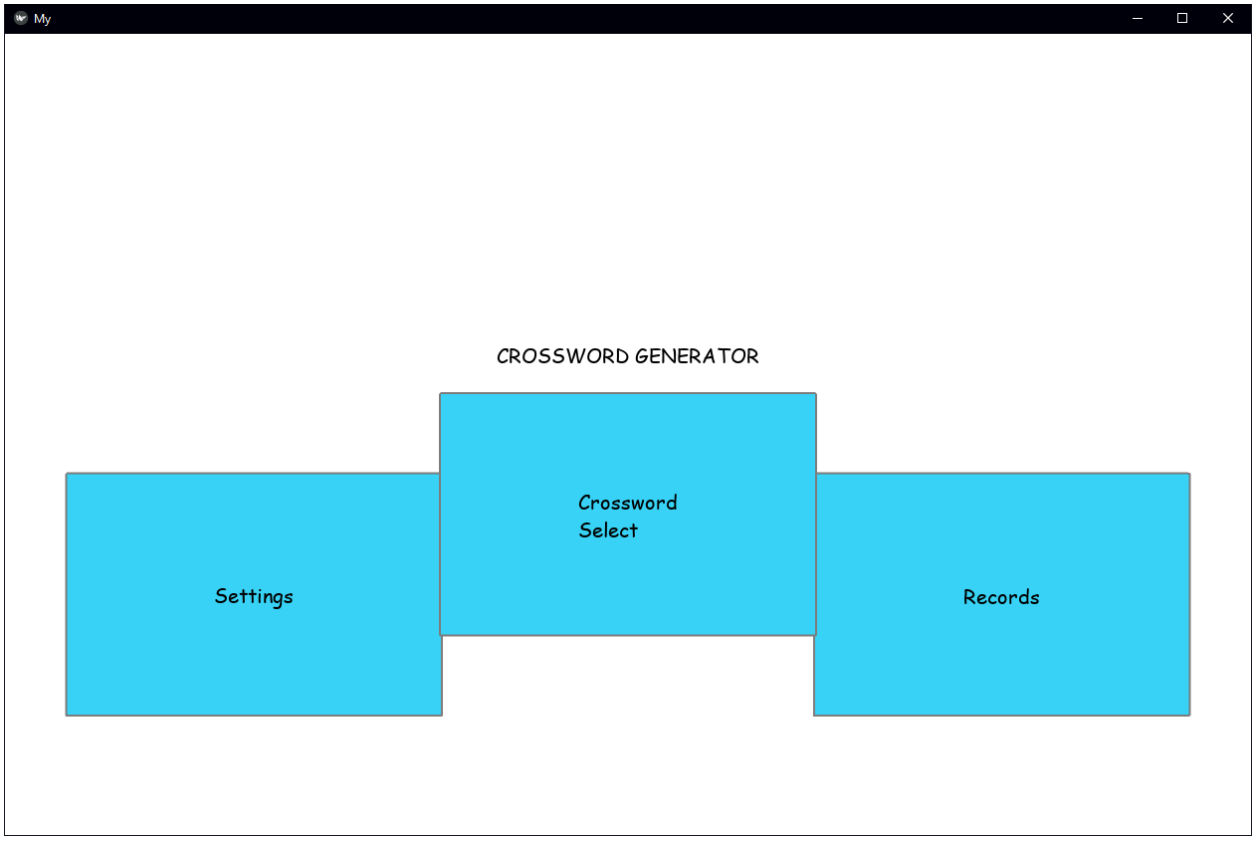
First Iteration:

Through the creation of this abstraction was applied by simplifying functions down into buttons which can be handled later on. The main objective was to link the pages together and relied heavily on the UI diagrams made in the design stage.

The exception to this was the Crossword Page itself. In order to even use Kivy to build my interface I needed it to be able to suit my projects demands. This involved displaying a dynamic grid which the user can update and take in text input. Text input was relatively straightforward to implement with Kivy's inbuilt functions but I had a lot of trouble generating the crossword grid itself. My program was able to generate the grid within the for loop but did not assign cells their IDs which were needed to read data and change colour etc.

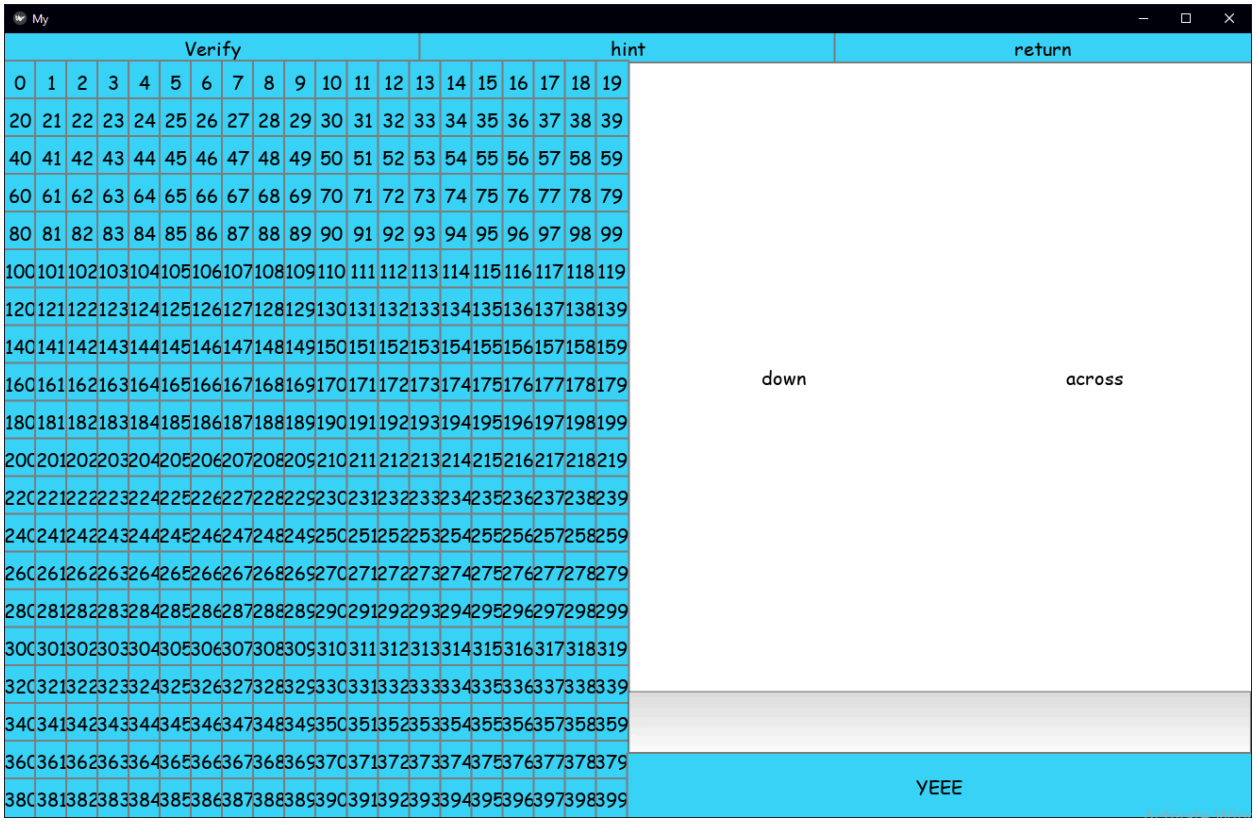
I asked online chat forums for help and one user got back to me, recommending to use ListProperty from kivy.properties to give the GridLayout itself the ID. Through this solution the crossword cells would be the children of GridLayout therefore meaning they could be referenced through the index of a ListProperty. This solution worked perfectly and I was able to continue development of the crossword.

There were no other major hurdles during this iteration and linking the screens themselves was simplistic.


My
CROSSWORD GENERATOR
Crossword
Select
Settings
Records

Home Page

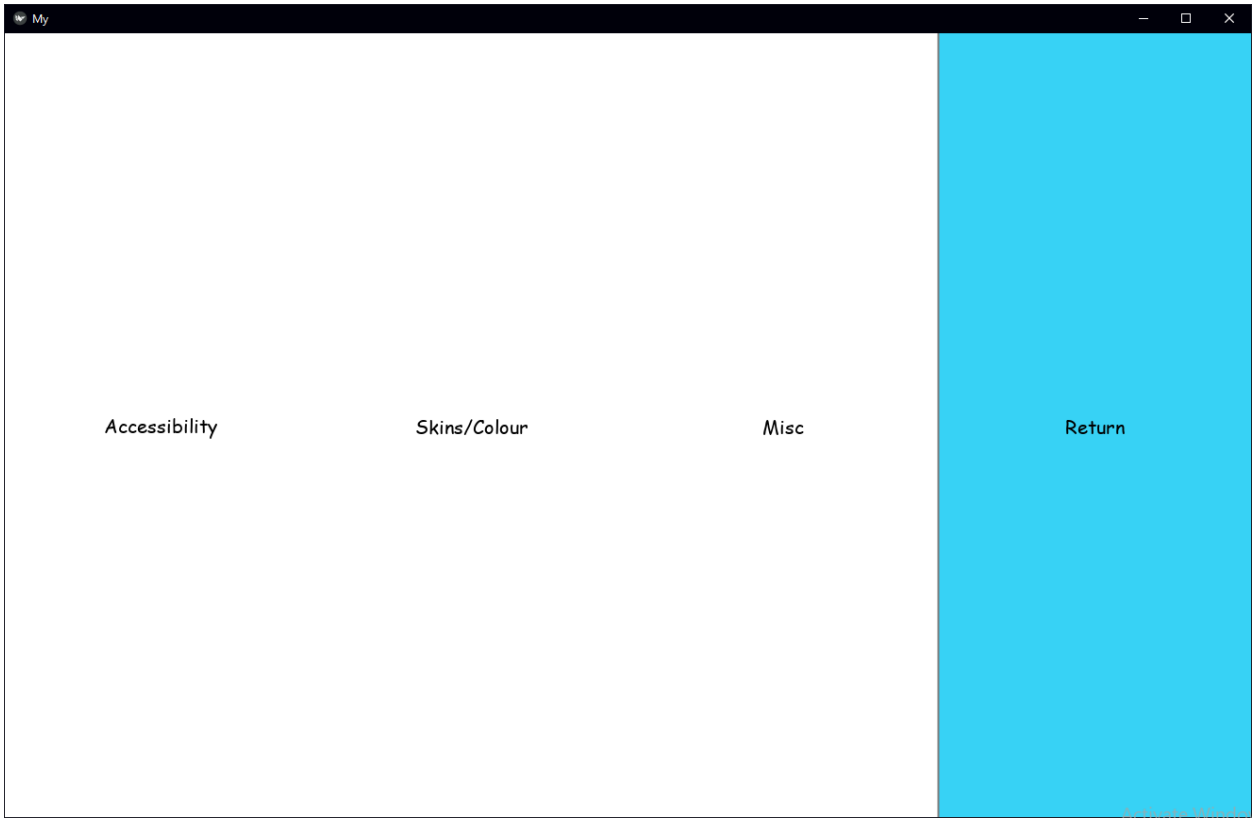
Difficulty select. Currently does not change difficulty variable



Crossword page. Cells's text is set to their index in the list. Buttons Verify and Hint will be given their respective functions later on in development

Testing that grid is responsive and text input works. Currently the program is configured to change the first 3 indexes of the grid when the button labelled “YEEE” is pressed

The grid's values have changed indicating a successful test. The colour of the text field has also changed. This was just testing to ensure colours can be altered alongside text.



The Settings page. Labels act as placeholders for the lists of buttons which will eventually take their place. Return takes the user to Home Page

Records Page. Has labels in place for the real data which will be implemented later. Reset data also has no function assigned yet which will also change.

Overall I am happy with this iteration as it sets the project up in a good spot for the next one.

Iteration 2:

The goal of this iteration was to add functionality to the buttons placed in the previous iteration and to make sure the data being passed is in the correct form.

I started with the crossword itself and made it so that it will correctly function (take inputs ,verify etc) when fed the correct data. Based on this I will have to slightly redesign my algorithm to give this data.

The colour of my UI is done by assigning “colour variables” such as “outercolour”,”gridcolor”,”flashcolor” etc a colour value.

When the program is run these colours are given to each ID in a fitting manner. Skins will work by changing these colour variables and therefore changing the outline colour, colour of the grid and so on. These can be toggled in the settings menu and testing will be done to ensure one cannot have more than one skin equipped at a time.

IDE:

Throughout development multiple IDEs were used for different purposes. Firstly I found Idle best for the letter linking functions due to how easy it was to customise the output window so that I could see exactly what was going on in arrays and where.

VSCode was used for the main bulk of functions other than this. I found the ease of access settings very helpful when dealing with so many different variables.

Finally I used PyCharm for the GUI development. I just found it most comfortable and easy to work on multiple files in the closed environment. It also has great module support and automatically downloads what you need after debugging.

Dictionary

After searching online for a list of english words with their definitions I found https://www.vocabulary.com/lists/52473
This website lists frequent but challenging words and even orders them by difficulty. I pasted all 1000 words into a file and then split them into quarters based on the website’s difficulty. After this was done I needed to format the file into how I wanted it. I only need the word, its definition and its difficulty.

```
import csv
toAdd = []
header = ['word','def']


with open("round4.txt") as file:
    with open("wordsSorted.csv","a",newline='') as csvfile:
        lines = file.readlines()
        writer = csv.writer(csvfile)
        #writer.writerow(['word','definition','difficulty'])
        for i in range(0,len(lines)):
            if lines[i][0] == " ":
                x = (lines[i].split("\n")[0]).split(" ")[1]
                y = lines[i+1].split("\n")[0]
                print(x[0]," : ",lines[i+1])
                toAdd.append(x)
                toAdd.append(y)
                toAdd.append('4')
                writer.writerow(toAdd)
                toAdd = []
```

I created this program to format the words.

They went from

incidental
not of prime or central importance
The models themselves are incidental on “Scouted,” merely empty planets around which revolve some fascinating characters and plenty more dull ones.New York Times (Nov 27, 2011)
acquiesce
agree or express agreement
American officials initially tried to resist President Karzai’s moves but eventually acquiesced.New York Times (Mar 9, 2012)
slew
a large number or amount or extent
In fact, intense focus may be one reason why so-called savants become so extraordinary at performing extensive calculations or remembering a slew of facts.Scientific American (Mar 3, 2012)
usurp
seize and take control without authority
More than anything, though, officials expressed concern about reigniting longstanding Mexican concerns about the United States’ usurping Mexico’s authority.New York Times (Mar 15, 2011)
sentinel
a person employed to keep watch for some anticipated event
The prisoners undressed themselves as usual, and went to bed, observed by the sentinel.Drake, Samuel Adams

To this

word,definition,difficulty
consider,deem to be,1
minute,infinitely or immeasurably small,1
accord,concurrence of opinion,1
evident,clearly revealed to the mind or the senses or judgment,1
practice,a customary way of operation or behavior,1
intend,have in mind as a purpose,1
concern,something that interests you because it is important,1
commit,"perform an act, usually with a negative connotation",1
issue,some situation or event that is thought about,1
approach,move towards,1

Once this was done I just had to thread them through the difficulty algorithm which will be done at a later stage.

LetterSort:

After further testing this function I noticed a design error in the case that it only functions properly for arrays which can link the words next to each other ie

[“gorilla”, “monkey”, “yes”]

However if the array were to be rearranged as

[“gorilla”, “yes”, “monkey”]

It would not work. This is because the algorithm works by index. It applies “gorilla” to the grid below as shown.


```
[['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', 'g', 'o', 'r', 'i', 'l', 'l', 'a', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*'],
['*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*']]
```


It will then try and apply “yes” on top of it indefinitely because “gorilla” and “yes” do not link. My solution is to implement a failsafe in the case of this happening that triggers after too many unsuccessful attempts. When triggered it will end the gridAdd() function and return data stating what happened. The program will then move the word which was going to be added to the end of the array, which is now acting as a circular queue.

The algorithm now looks like this

```
for i in range(1,len(words)):
    checked = False
    while checked == False:
        newgrid,checked,start = gridAdd(words[i][0],grid)
        if checked == False:
            words.append(words.pop(i))
    grid = newgrid
    start.append(words[i][1])
    directions.append(start)
```

While designing this failsafe I realised there must also be one for if the words simply cannot be linked and a new set is in order. This will trigger in the loop of the algorithm above, when all of the words remaining in the list have been pushed to the back of the queue.

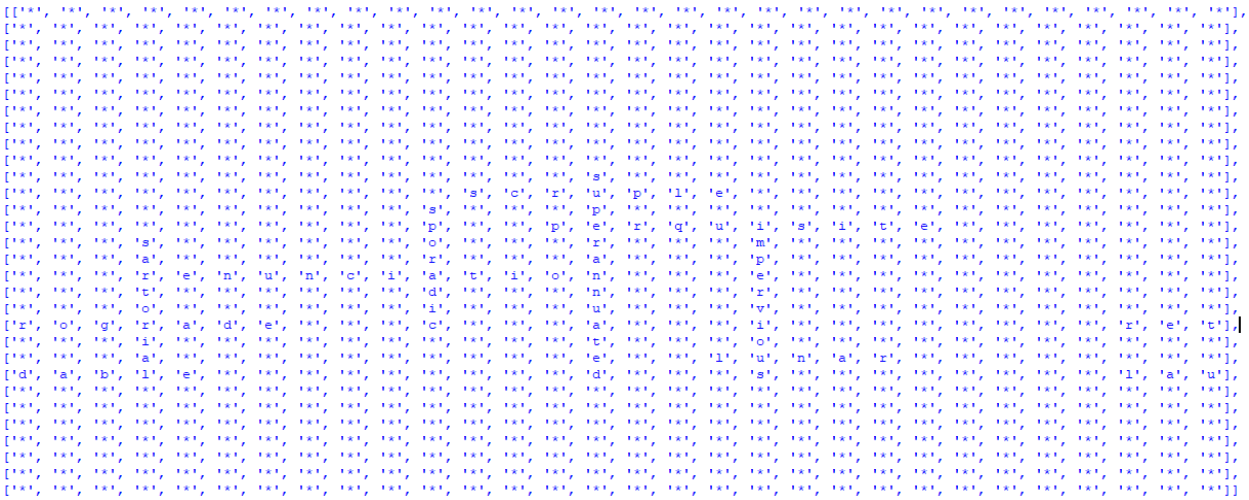
```
for i in range(1,len(words)):
    checked = False
    k=0
    while checked == False:

        newgrid,checked,start = gridAdd(words[i][0],grid)
        if checked == False:
            words.append(words.pop(i))
            k=k+1
            if k > len(words):
                print("\n\n\n\n\n\n\n\FAIL")
```

Currently there is no function to run when this happens but at least for now I know when it triggers.

```
TESTING:
*  :  i
*  :  c
*  :  i
*  :  d
*  :  u
*  :  j
o  :  o
u  :  u
*  :  s
pre: *
post: *
PAASSEED
```

Letter link also needs to return the grid position of the start of each word.

Another issue I encountered while experimenting on larger data sets was the program



In this case both retrograde and laudable carry across from the left side to the right side and the program passes it as a success, which we don't want.

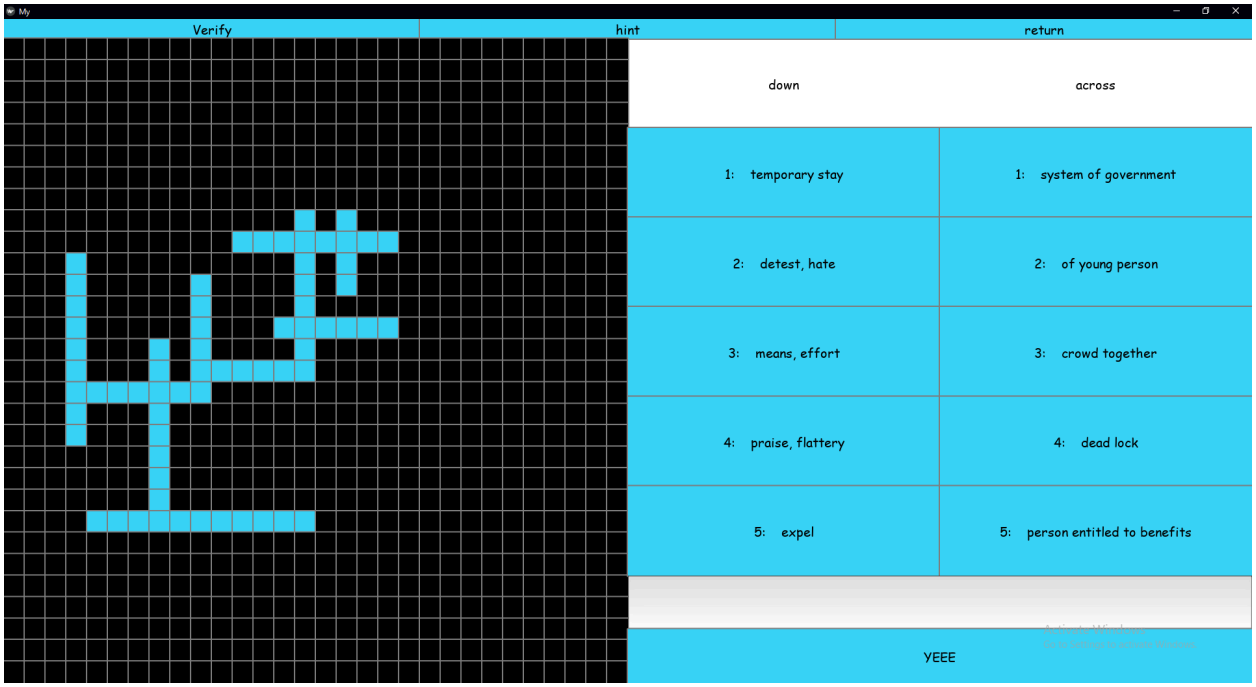
My initial proposition was to add another failsafe for if the program tried adding words of certain length too close to the edge. This would be made accurate by the pre and post variables so it would be possible to still have words correctly on the border

For this situation but happening vertically I implemented a try, except for validation. This fits as a solution in this case because the program simply crashes because the array is too small.

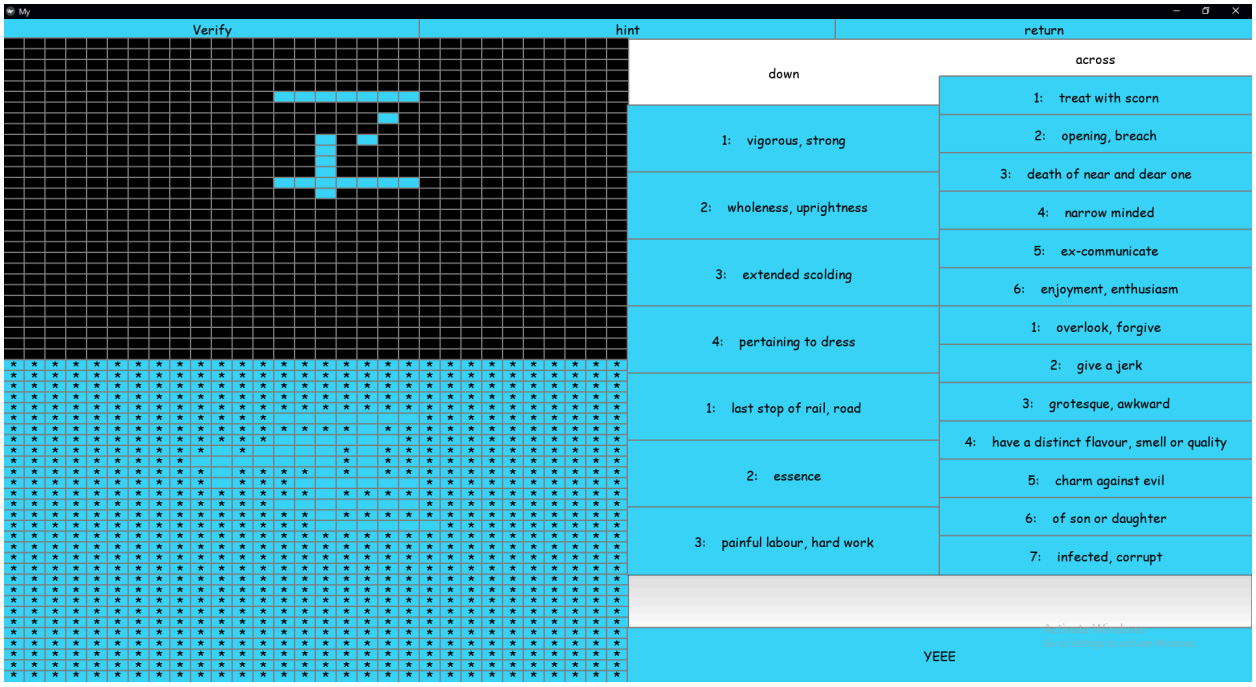
```
if len(pre) > position[0] or len(post)+position[0]>29:
    horizontal = False
    print("FAILSAFE")
```

```
==========================================================
immick
TESTING:
FAILSAFE
*  :  g
gimm ick
ick
```

After implementing the letterLink and wordSelect functions into my UI the crossword page functioned as expected upon selecting a difficulty

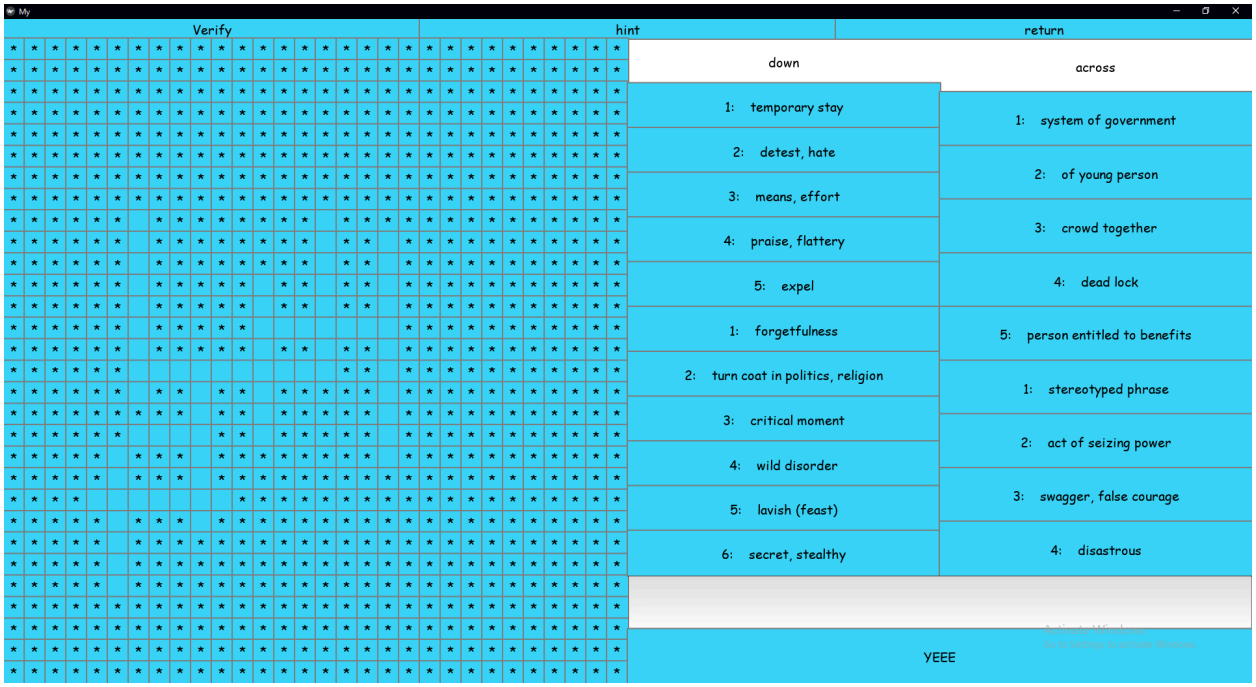


However when pressing return to go to the homepage and then navigating back to the crossword page not all of the old grid would be replaced.



My solution to this was to remove all existing widget children of the grid upon generating a new crossword.

```
def on_pre_enter(self, *largs):
    self.ids.thegrid.clear_widgets(children=None)
```



This is the result of doing so. Upon inspection the clear spaces in the grid are in line with the latest generated words however they are not formatted correctly and do not respond to the text box and highlight functions. The clues also show the clues from the crossword before, evidencing that it has simply been appended to.

My initial thought was to repeat what I did with the grid's children but to the clues and to reimplement the formatting.

```
def on_pre_enter(self, *largs):
    self.ids.thegrid.clear_widgets(children=None)
    self.ids.downclues.clear_widgets(children=None)
    self.ids.acrossclues.clear_widgets(children=None)
```

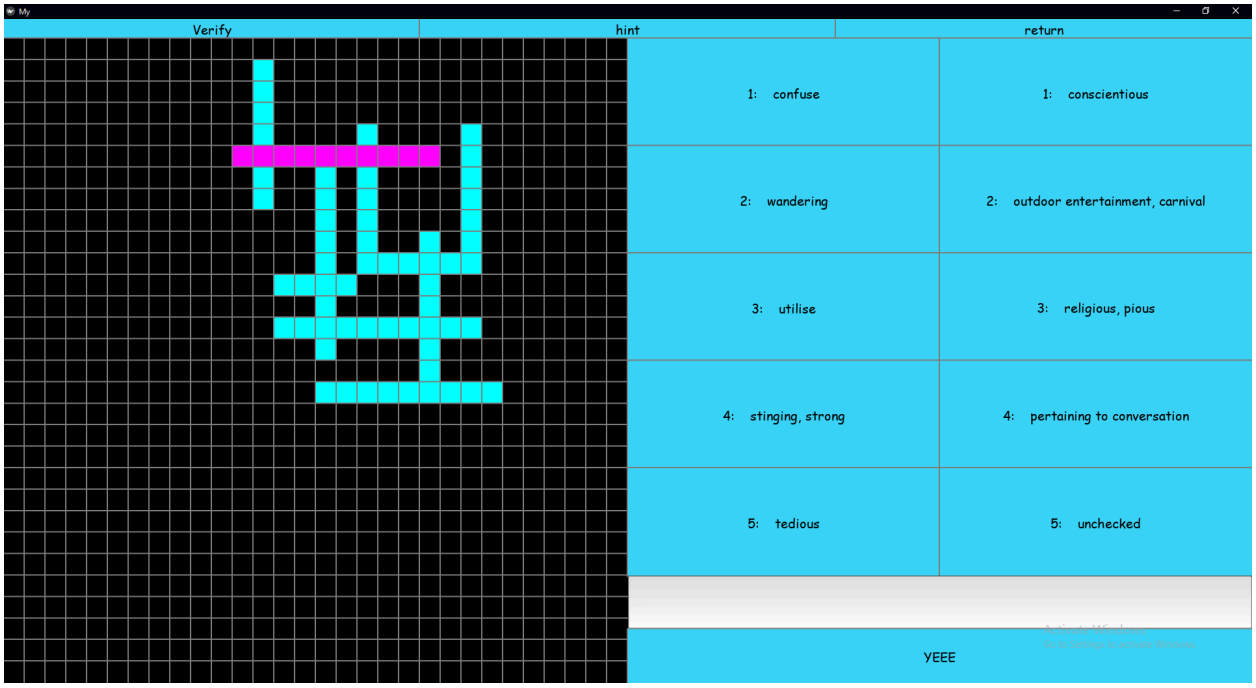
Doing this fixed the clues from keeping old records.

```
582 9
 :  c
o :  o
n :  n
 :  s
e :  e
n :  n
s :  s
* :  u
s :  s
```

Upon investigation I discovered that the hint button successfully changed the answer values seen through verification

```
def on_pre_enter(self, *largs):
    self.ids.thegrid.clear_widgets(children=None)
    self.ids.downclues.clear_widgets(children=None)
    self.ids.acrossclues.clear_widgets(children=None)
    self.my_buttons = []
    global selected
    global lastchanged
    selected = [-1,-1,"no"]
    lastchanged = "null"
```

These are the changes I made to make the function work as intended. Variables selected and lastchanged needed to be reset for the next potential crossword.



Text Input

In order to add to the crossword the user must type in the textbox and either press the button or press the enter key.

I created the following function to assign the inputs to the grid.

```
def crossword_append(self):
    print(self.lastSelected[1])
    if len(self.text_input.text) == (self.lastSelected[1]):
        if self.lastSelected[2] == "across":
            for j in range(self.lastSelected[1]):
                self.my_buttons[self.lastSelected[0] + j].text = self.text_input.text[j]
        if self.lastSelected[2] == "down":
            for j in range(self.lastSelected[1]):
                self.my_buttons[self.lastSelected[0] + 30 * j].text = self.text_input.text[j]
```

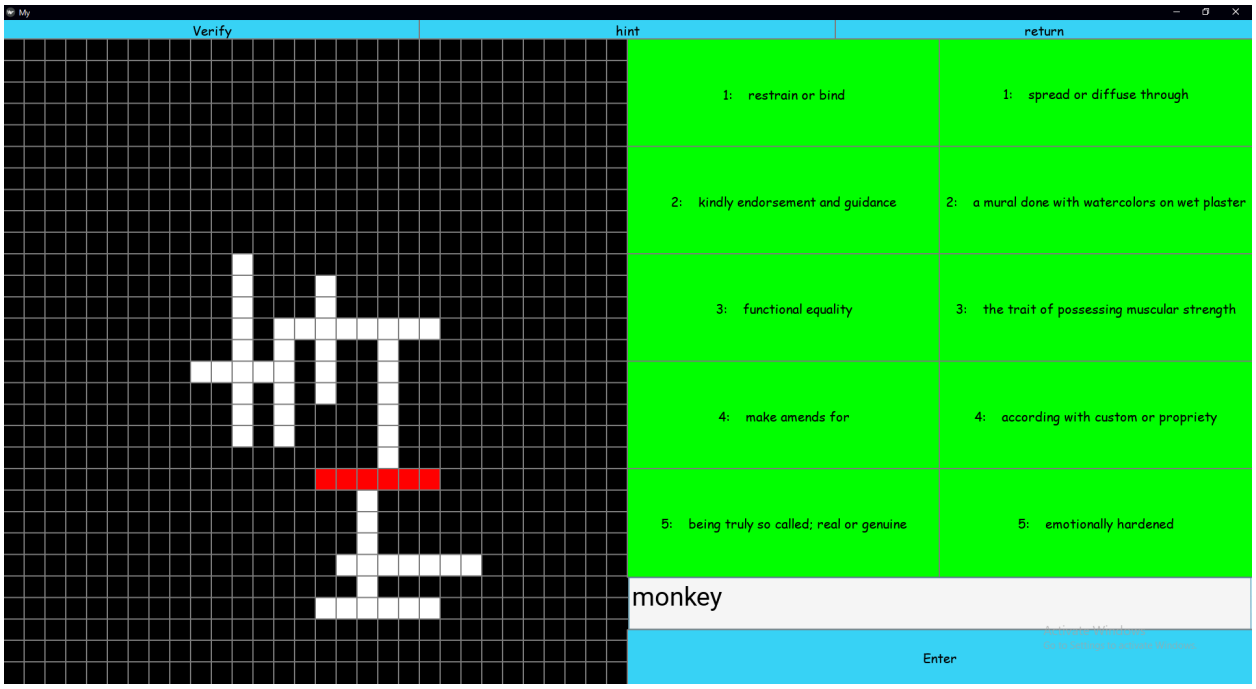
It works by using the data from the lastSelected array in order to determine where the spaces which need to be filled are stored. Depending on whether lastSelected is a down or across clue the program either iterates in intervals of 1 or 30 in order to go to the next column or row. Minor validation is done here to ensure the user can only input words of the same length as the grid space selected.

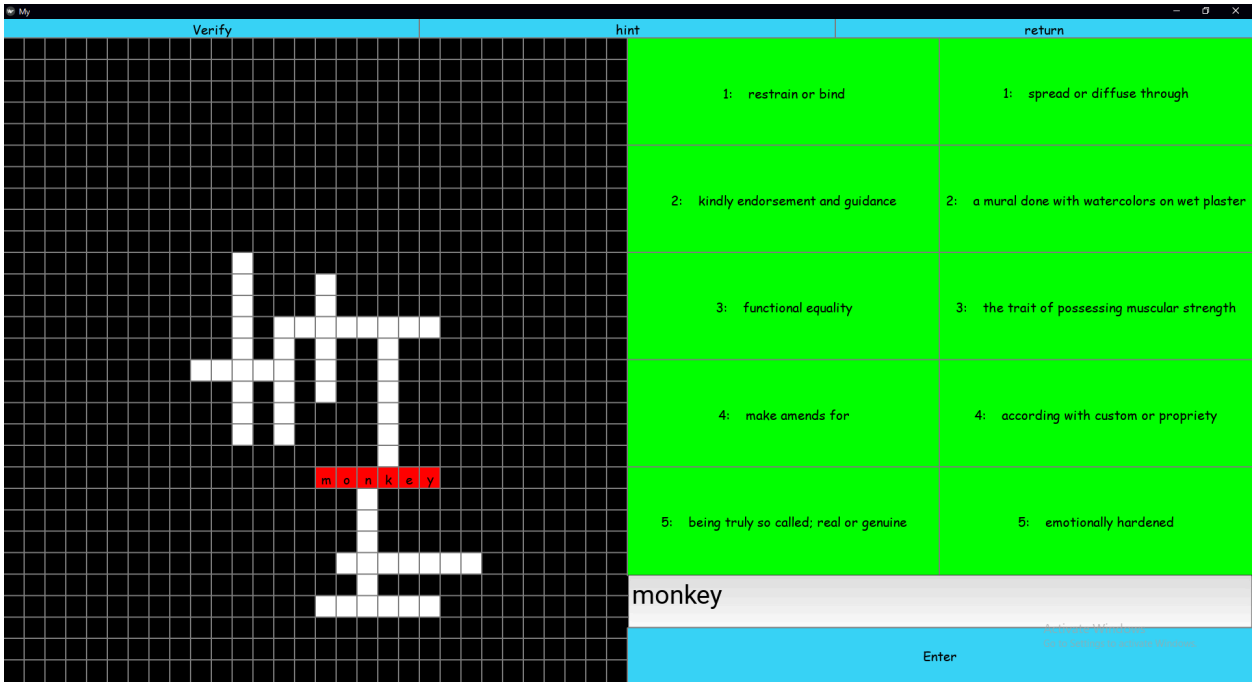
```
if len(self.text_input.text) == (self.lastSelected[1]) and self.text_input.text.isalpha():
```

I further improved this by adding an isalpha statement as the data required to be inputted will never be anything but letters.

For the following tests, the input box tints slightly grey after a word is inputted so that it is possible to know when an input has been attempted

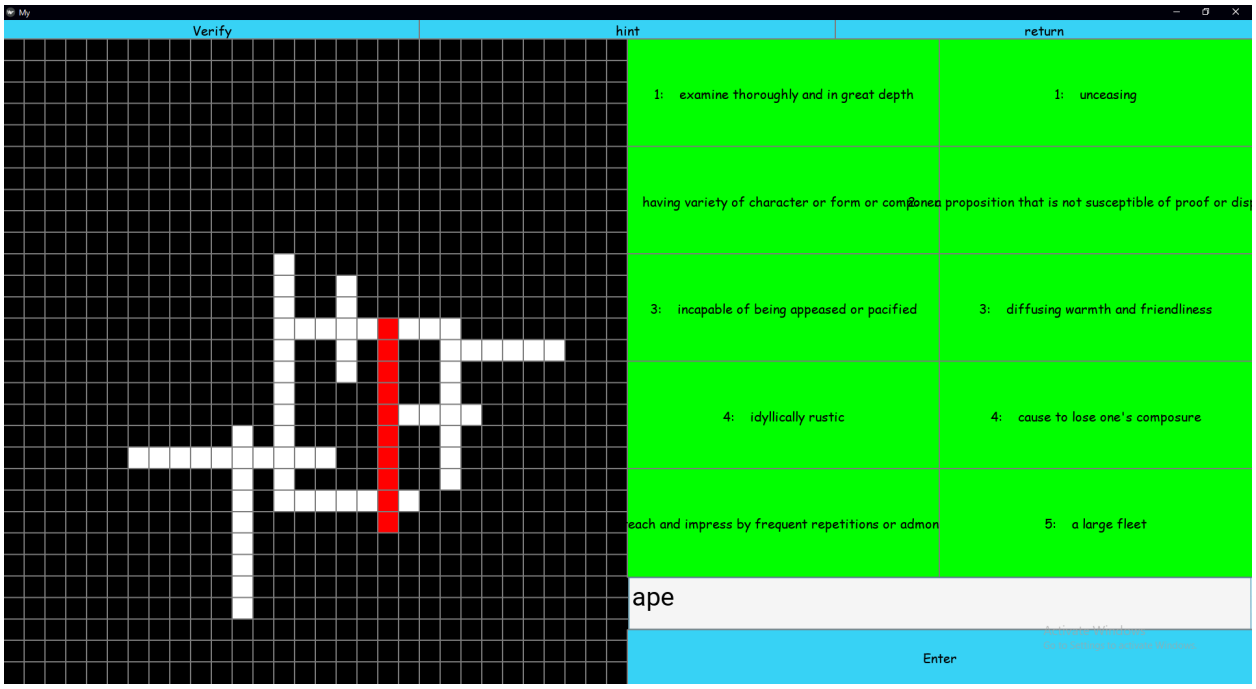
Test ID 01

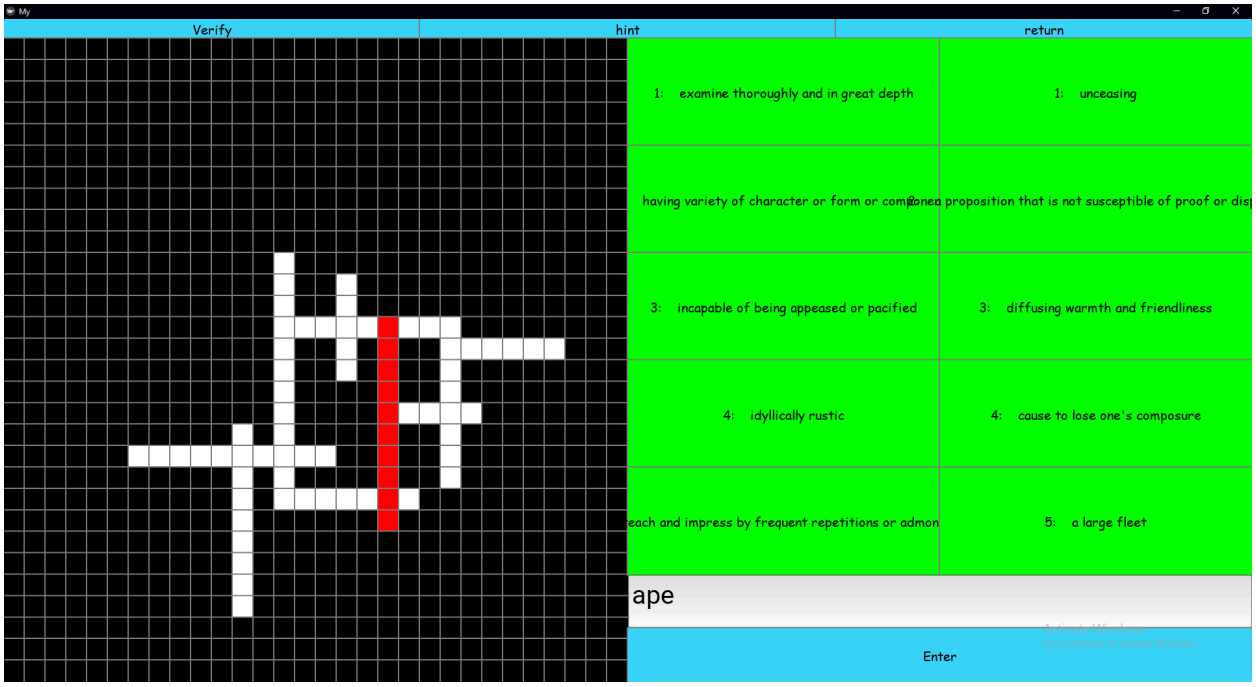




success

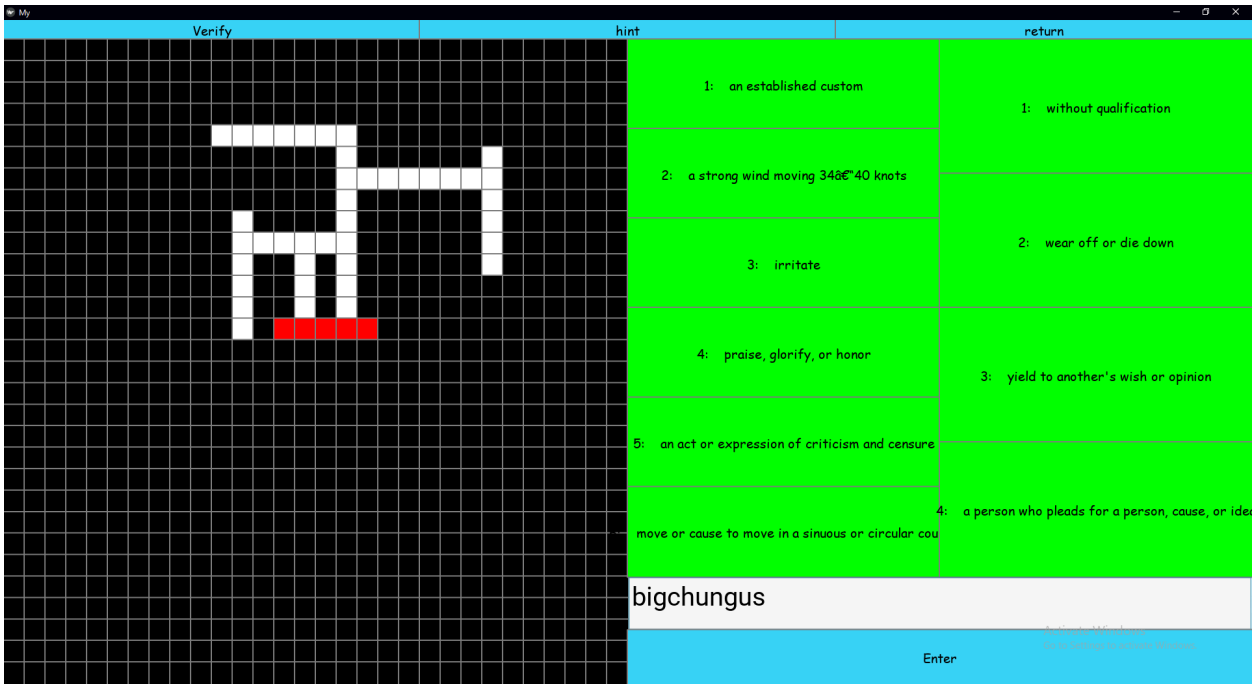
Test ID 02

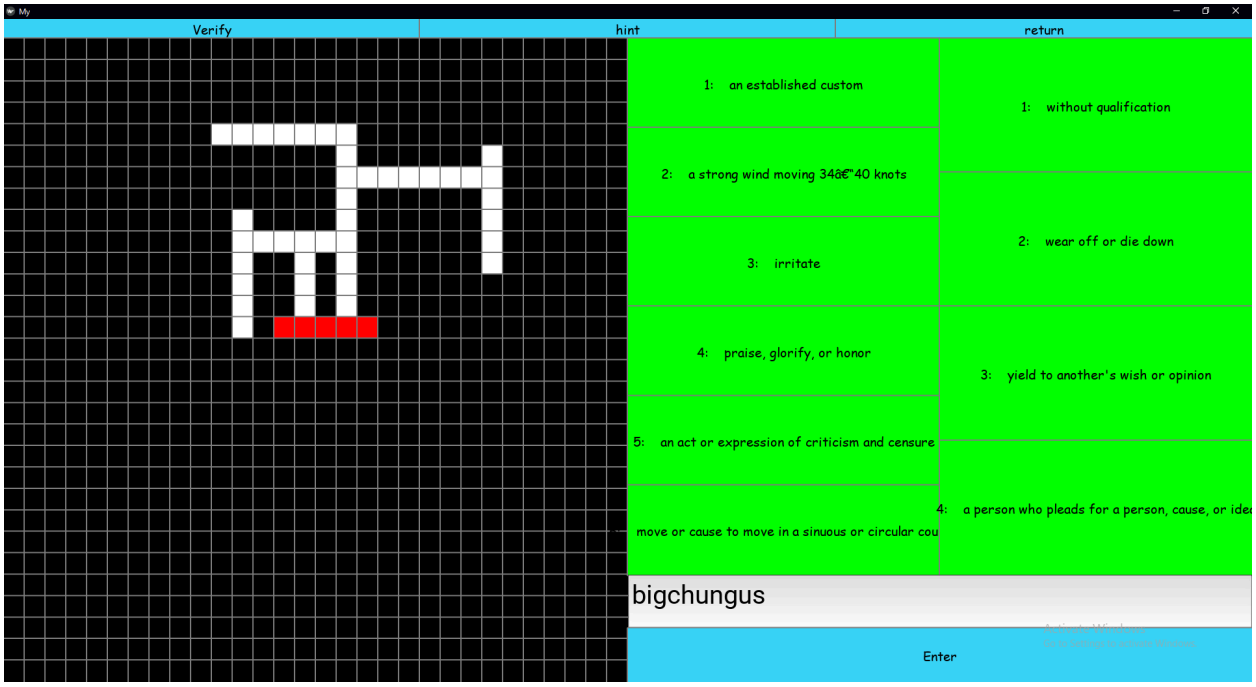




Refuses input, success

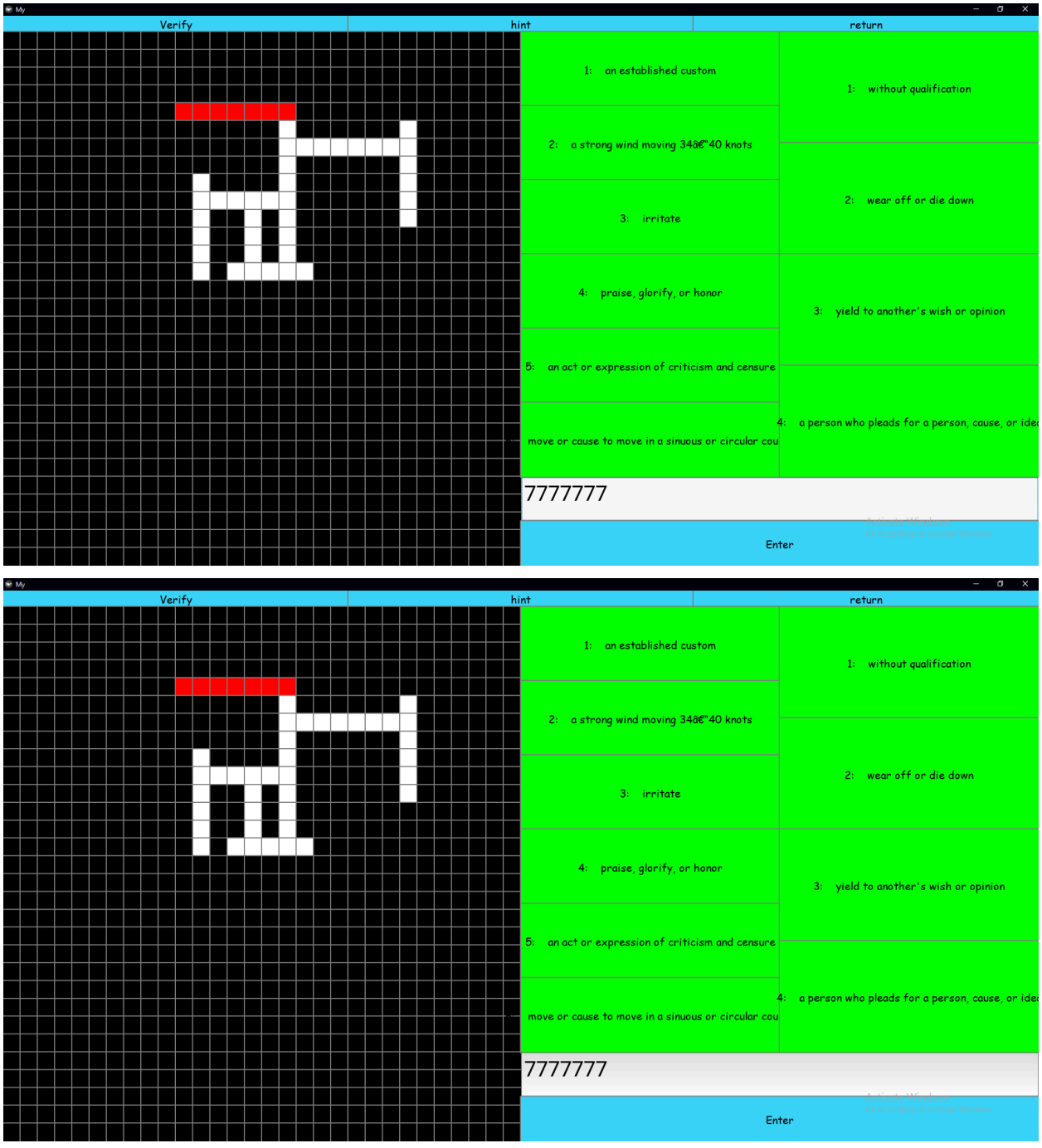
Test ID 03





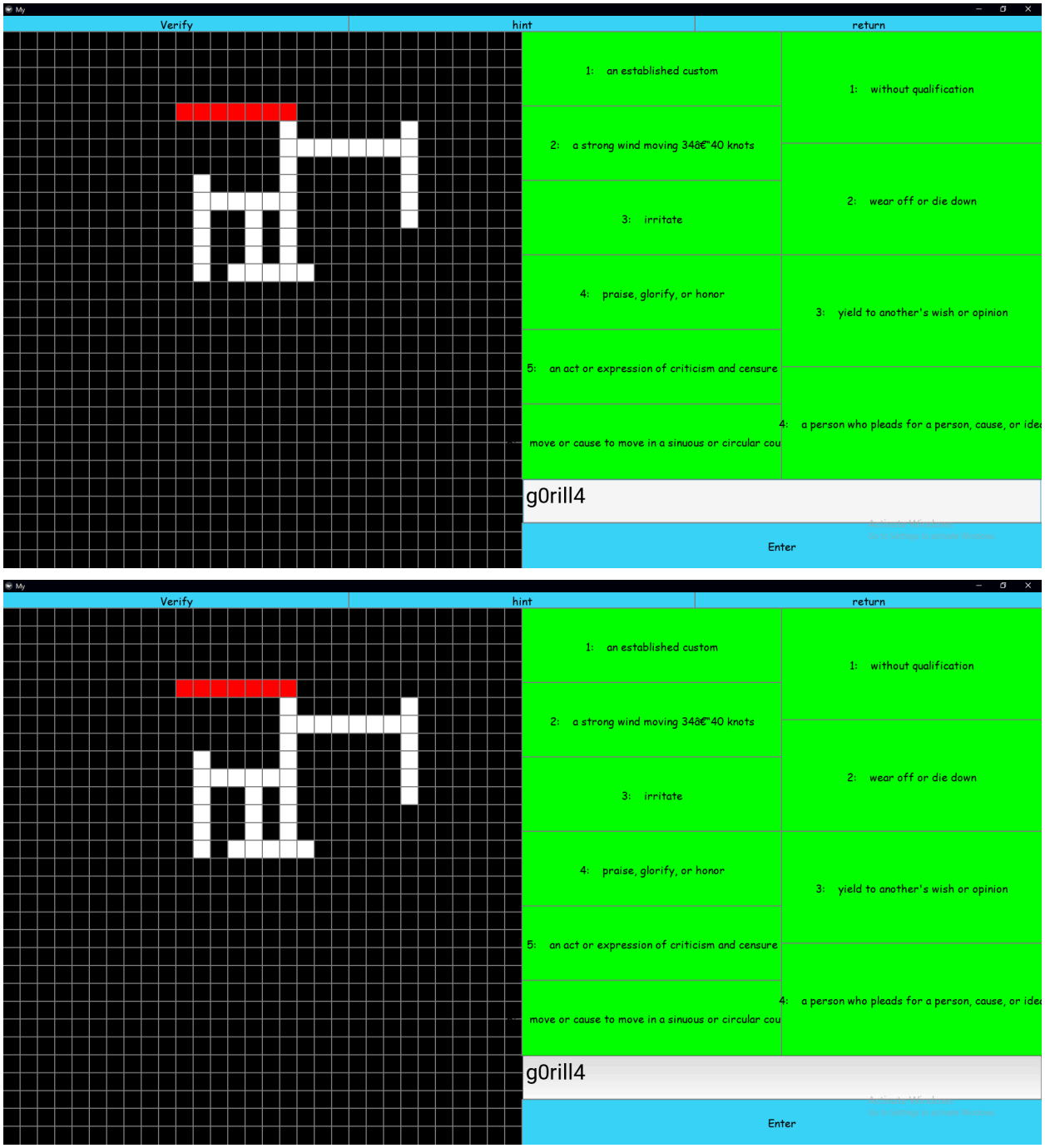
Refused input, success

Test ID 04



Refused input, success

Test ID 05



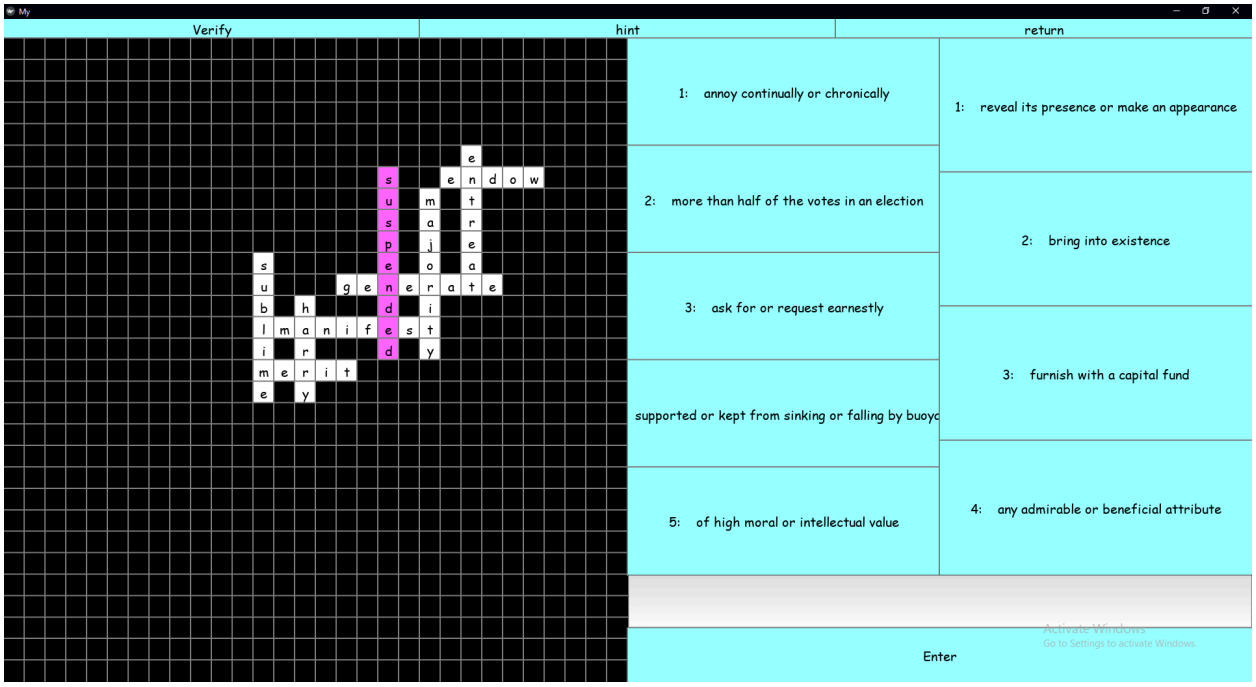
Rejected successfully

Difficulty Select:

The objective of these buttons is to pass on the difficulty selected to crossword_generate and continue to the difficulty screen.

Test ID 10

Check that easy button can be pressed and creates a crossword of easy words
Expected crossword. Reference file to see if easy words

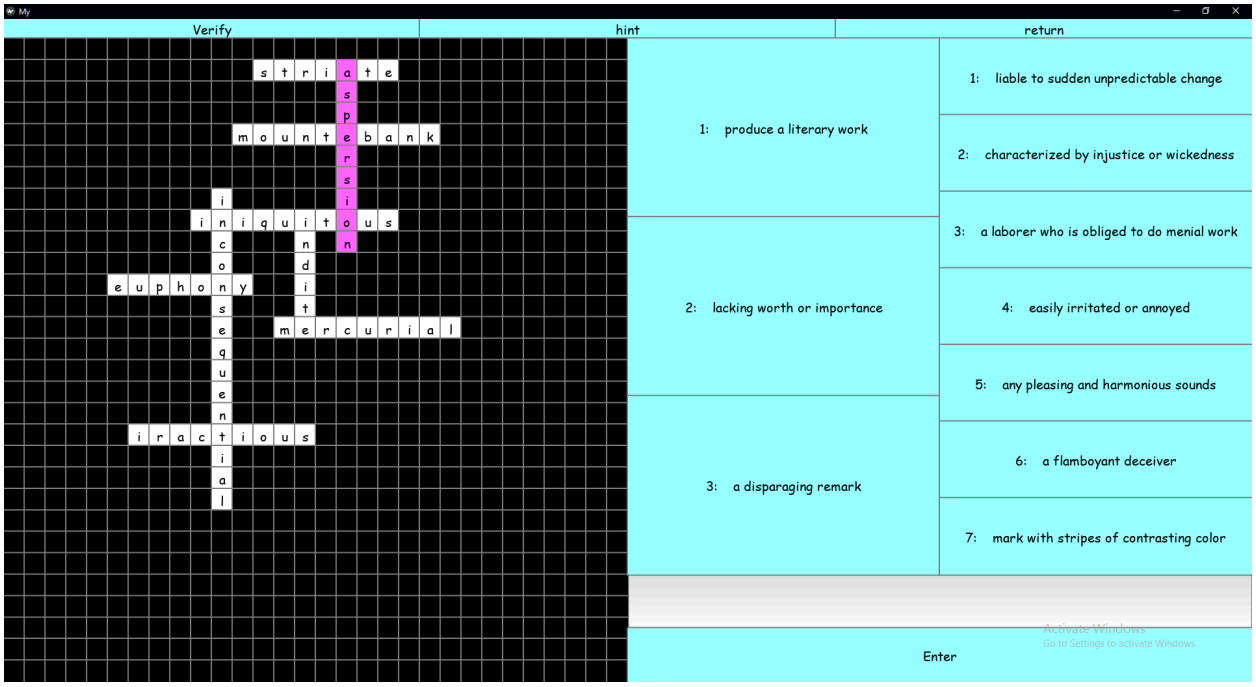



```
sublime,of high moral or intellectual value,1
endow,furnish with a capital fund,1
merit,any admirable or beneficial attribute,1
```

Runs correctly and only uses difficulties of 1 (easy), success

Test ID 11

Check that extreme button can be pressed and will create a crossword of extreme words.




```
iniquitous,characterized by injustice or wickedness,4
mercurial,liable to sudden unpredictable change,4
inconsequential,lacking worth or importance,4
```

Crossword generated and all of difficulty 4 (extreme). Success

First Time Setup

At this point in development I'd already decided what data I wanted to record for the stattrack and implemented them into first_time_setup() with appropriate initial data.

```
import csv
def first_time_setup():   #Setup stattrack values, unlocks, etc
    with open('scores.csv','w',newline='') as csvfile:
        writer = csv.writer(csvfile)
        writer.writerow(["Total Time",0])
        writer.writerow(["Average Time",0])
        writer.writerow(["Total Words",0])
        writer.writerow(["Total Letters",0])
        writer.writerow(["Total Crosswords",0])
        writer.writerow(["Easy Completions",0])
        writer.writerow(["Medium Completions",0])
        writer.writerow(["Hard Completions",0])
        writer.writerow(["Extreme Completions",0])
        writer.writerow(["Fastest Easy",-1])
        writer.writerow(["Fastest Medium",-1])
        writer.writerow(["Fastest Hard",-1])
        writer.writerow(["Fastest Extreme",-1])
        csvfile.close()
```

For all cumulative values the value is set to zero. The fastest time for each difficulty is set to -1 so that when comparing for a new fastest difficulty a check will be done to see if time is negative which will allow it to be overwritten without question. The will also be measures in place to ensure "N/A" is displayed on the Records page if -1 is still the value.

Average time is not cumulative however by storing the total number of crosswords we can calculate it by doing (total time/total crosswords).

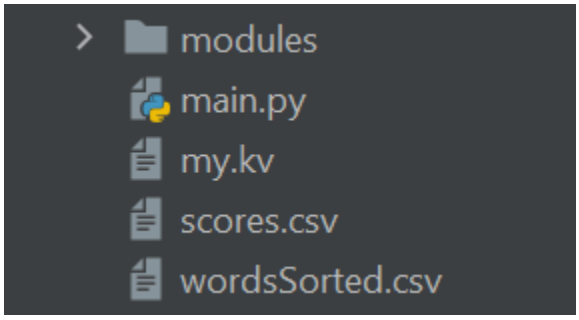
This procedure is only supposed to be run upon the first time the application is run. In order to do this another procedure is needed to check if the file already exists.

I designed on_start() for this purpose

```
def on_start():
    try:
        with open('scores.csv', 'r', newline='') as csvfile:
            pass
    except:
        first_time_setup()
```

It functions by using a try, except statement to check if the scores file already exists. If it does exist then the program is left to run as normal. If it excepts then first_time_setup is run. It will also run in the case that the file is missing which is an unintended benefit.

Upon running the program for the first time, scores.csv is expected to be created



```
Total Time,0
Average Time,0
Total Words,0
Total Letters,0
Total Crosswords,0
Easy Completions,0
Medium Completions,0
Hard Completions,0
Extreme Completions,0
Fastest Easy,-1
Fastest Medium,-1
Fastest Hard,-1
Fastest Extreme,-1
```

The file was correctly generated in the correct domain, successful test

StatTrack

The stattracking algorithm also had to be updated to match the file formatting.

```
import csv


def stattrack(difficulty, timetaken, wordsanswered, lettersanswered):
    stats = []
    print(difficulty)
    with open('scores.csv', newline='') as csvfile:
        reader = (csv.reader(csvfile.readlines()))
        for line in reader:
            stats.append(line)
        stats[0][1] = int(stats[0][1]) + int(timetaken)
        stats[2][1] = int(stats[2][1]) + wordsanswered
        stats[3][1] = int(stats[3][1]) + lettersanswered
        stats[4][1] = int(stats[4][1]) + 1
        stats[4 + int(difficulty)][1] = int(stats[4 + int(difficulty)][1]) + 1
        stats[1][1] = int(stats[0][1]) / int(stats[4][1])
        if 0 < timetaken < int(stats[8 + int(difficulty)][1]):
            stats[8 + int(difficulty)][1] = int(timetaken)

    with open('scores.csv', 'w', newline='') as csvfile:
        writer = csv.writer(csvfile)
        for i in stats:
            writer.writerow(i)
```

This algorithm works by reading the existing stats file for data and storing it inside an array. Cumulative values passed in are added to their respective counterparts in the array and saved. The difficulty selected determines which difficulty_completions is used for this too. The same is done to find the index of the fastest difficulty which is then compared and handled accordingly. Average time is calculated based on the new values of total time and crosswords completed.

The file is then opened again and replaced with this new array.

Test ID 12
This test is to ensure that the non-time related values are stored correctly.

Test ID 13

Checks that the average and cumulative time taken are recorded properly

An error occurred in the above because stats[1][1] (average time) was not being stored as an integer but left as a float. This likely occurred due to me misreading the int statements in the calculation and assuming I'd already converted the entire value.

```
stats[1][1] = int(int(stats[0][1]) / int(stats[4][1]))                    #Average time
```

Adding an int at the start here fixed this and upon testing with 3 crossword completions the file reads:

```
Total Time,19
Average Time,6
Total Words,30
Total Letters,229
Total Crosswords,3
Easy Completions,2
Medium Completions,0
Hard Completions,1
Extreme Completions,0
Fastest Easy,3
Fastest Medium,-1
Fastest Hard,10
Fastest Extreme,-1
```

There was also an issue with the fastest time comparison which meant the value would never be updated because on the first_time_setup file it would read:

Test ID 14

Checks that a new time on a blank file will always be added.

IF 0 < timetaken < -1

To fix this I rewrote the statement as follows

```
if timetaken < int(stats[8 + int(difficulty)][1]) or int(stats[8 + int(difficulty)][1]) == -1:
    stats[8 + int(difficulty)][1] = int(timetaken)  #Respective difficulty fastest completion
```

This reiteration will always assign a new fastest difficulty if the base one had not been overwritten.

```
Total Time,5
Average Time,5
Total Words,10
Total Letters,71
Total Crosswords,1
Easy Completions,0
Medium Completions,0
Hard Completions,0
Extreme Completions,1
Fastest Easy,-1
Fastest Medium,-1
Fastest Hard,-1
Fastest Extreme,5
```

^^ File updating correctly

Test ID 15
Checks that a slower time is replaced by a faster time upon completion

Before, fastest time of 6

```
Total Time,6
Average Time,6
Total Words,10
Total Letters,72
Total Crosswords,1
Easy Completions,0
Medium Completions,1
Hard Completions,0
Extreme Completions,0
Fastest Easy,-1
Fastest Medium,6
Fastest Hard,-1
Fastest Extreme,-1
```

After, fastest time of 3

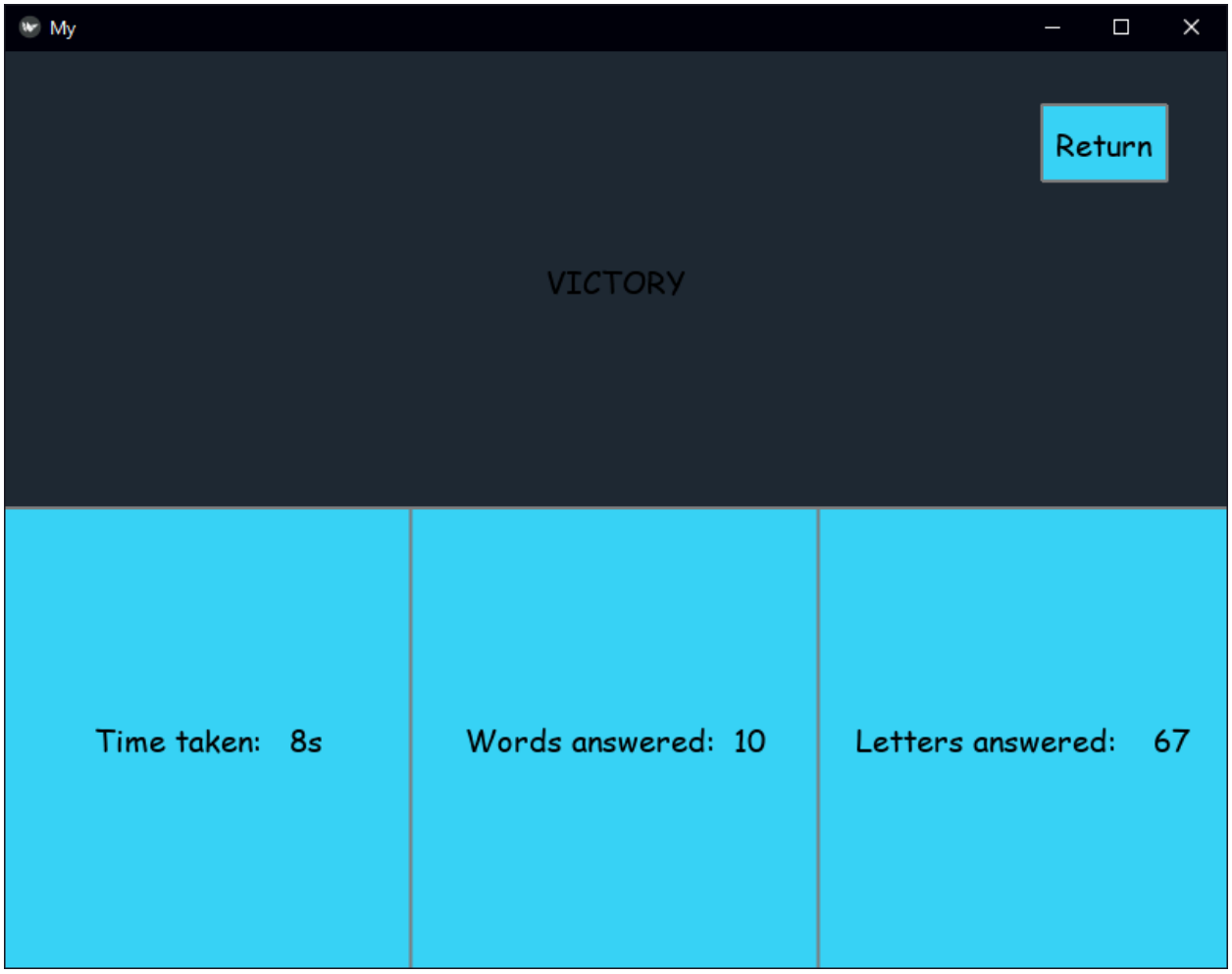
```
Total Time,9
Average Time,4
Total Words,20
Total Letters,151
Total Crosswords,2
Easy Completions,0
Medium Completions,2
Hard Completions,0
Extreme Completions,0
Fastest Easy,-1
Fastest Medium,3
Fastest Hard,-1
Fastest Extreme,-1
```

Successfully overwrites the slower time.

Test ID 16

Checks that a slower time does not replace a faster time upon completion

```
Total Time,9
Average Time,4
Total Words,20
Total Letters,151
Total Crosswords,2
Easy Completions,0
Medium Completions,2
Hard Completions,0
Extreme Completions,0
Fastest Easy,-1
Fastest Medium,3
Fastest Hard,-1
Fastest Extreme,-1
```



After, time of 8s newly recorded

```
Total Time,17
Average Time,5
Total Words,30
Total Letters,218
Total Crosswords,3
Easy Completions,0
Medium Completions,3
Hard Completions,0
Extreme Completions,0
Fastest Easy,-1
Fastest Medium,3
Fastest Hard,-1
Fastest Extreme,-1
```

The faster time is not overwritten as expected. Successful

Records Page

The records need to be read from the pre-existing csv file scores.csv.

```
def on_pre_enter(self, *args):
    stats = []
    with open('scores.csv', newline='') as csvfile:
        reader = (csv.reader(csvfile.readlines()))
        for line in reader:
            if line[1] == "-1":
                line[1] = "N/A"
            stats.append(line)
    for i in range(0, 5):
        self.ids.cumulative.children[4 - i].text = stats[i][0] + ":  " + stats[i][1]
    for i in range(0, 8):
        self.ids.individual.children[7 - i].text = stats[5 + i][0] + ":  " + stats[5 + i][1]
```

The code above reads the scores.csv file and creates an array of the values in it at index 1. As mentioned in the first time setup documentation a value of -1 will be set to “N/A” for the fastest recorded times as this saves comparing data types during the stattrack function. There are also 2 for loops here, one for each column of the records page.

on_pre_enter(self) is a kivy function which causes the below code to be executed when that page is navigated to. This is done so that users can leave the page, complete a crossword, return to the page and observe their new stats.

Conversion

Part of the success criteria was to have the timed data be validated and be displayed in hours, minutes and seconds. To do this I created a function named time_convert to convert the raw time in seconds to a more readable hour, minute, time format.

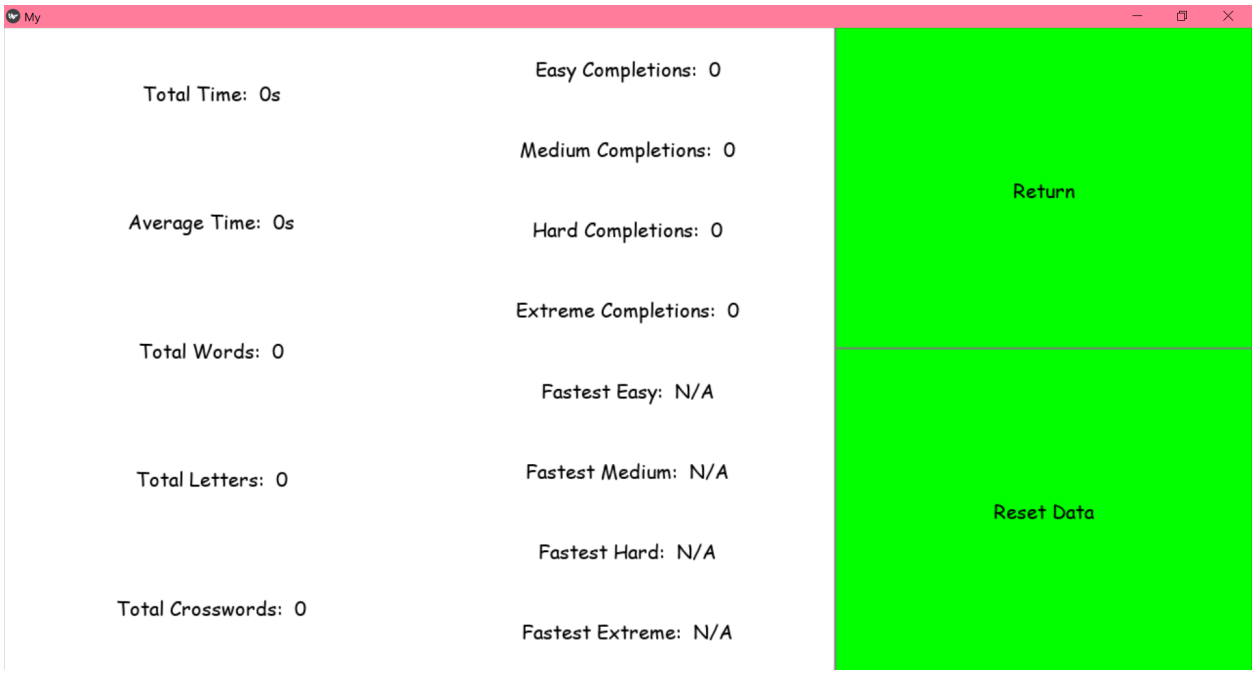
```python
def time_convert(time_value):
    print(time_value)
    minutes = time_value // 60
    hours = minutes // 60

    if minutes > 0:
        if hours > 0:
            converted = (str(hours) + "h, " + str(minutes % 60) + "m, " + str(time_value % 60) + "s")
        else:
            converted = (str(minutes) + "m, " + str(time_value % 60) + "s")
    else:
        converted = (str(time_value) + "s")

    return converted
```

The algorithm utilises python's floor division and modulus operators to correctly convert the time. It is used in the Records page and the Victory page to display results

Test ID 17

Check that any unrecorded times are displayed as N/A.

```
Total Time,0
Average Time,0
Total Words,0
Total Letters,0
Total Crosswords,0
Easy Completions,0
Medium Completions,0
Hard Completions,0
Extreme Completions,0
Fastest Easy,-1
Fastest Medium,-1
Fastest Hard,-1
Fastest Extreme,-1
```



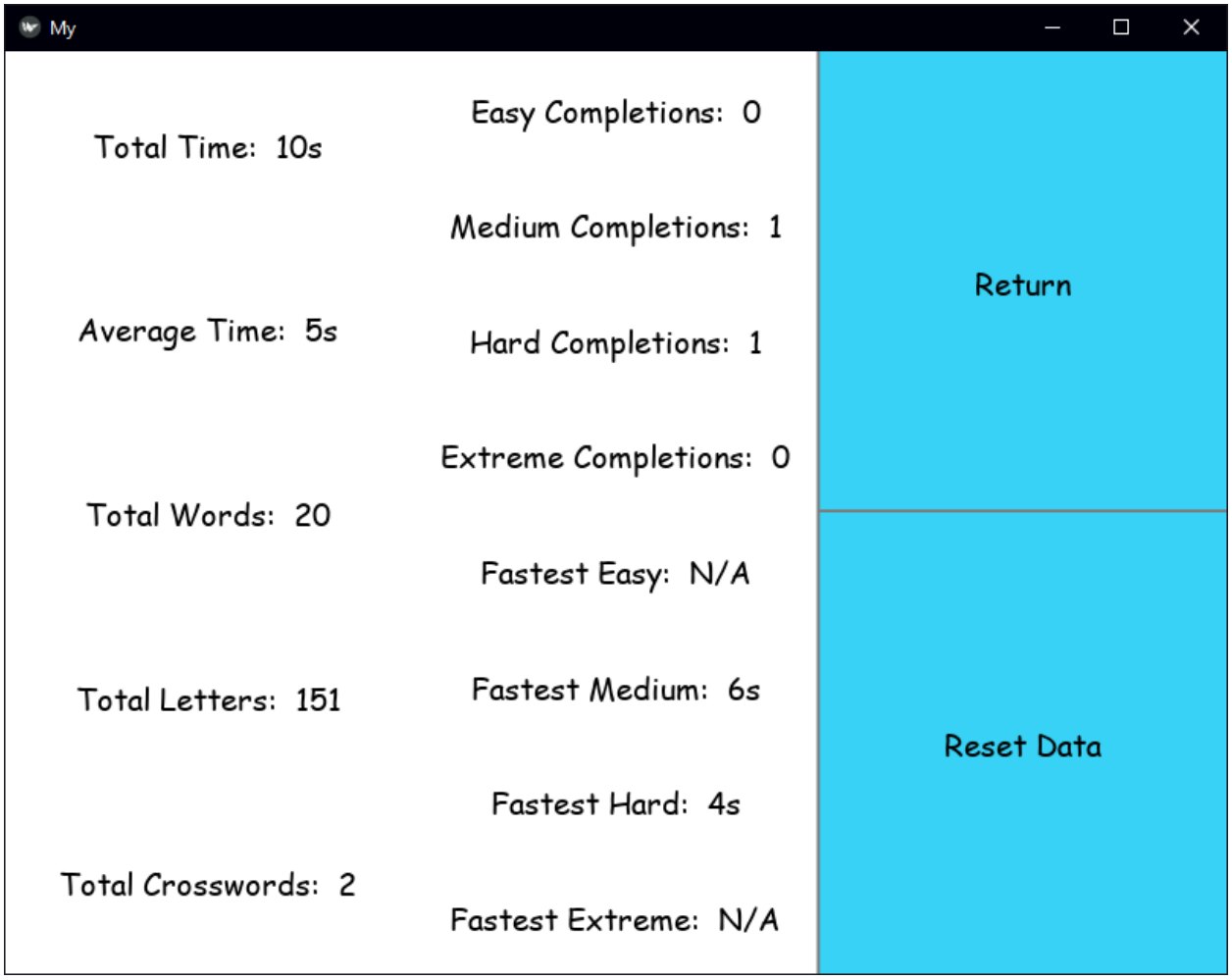
Displays the fastest times as “N/A” and everything else as zero as expected: success. One might argue that average time should be “N/A” if no crosswords have been completed but if the based on other applications I use which format the same way they do it the same way as I have.

Test ID 18 & 19
Check that scores.csv values are displayed correctly on the records page

The file data:

```
Total Time,10
Average Time,5
Total Words,20
Total Letters,151
Total Crosswords,2
Easy Completions,0
Medium Completions,1
Hard Completions,1
Extreme Completions,0
Fastest Easy,-1
Fastest Medium,6
Fastest Hard,4
Fastest Extreme,-1
```

Now produces

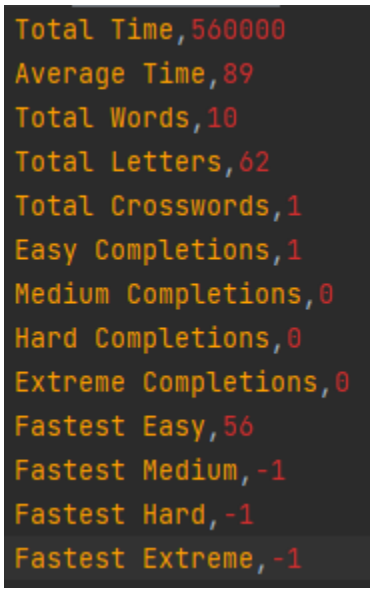


As an output. Successful

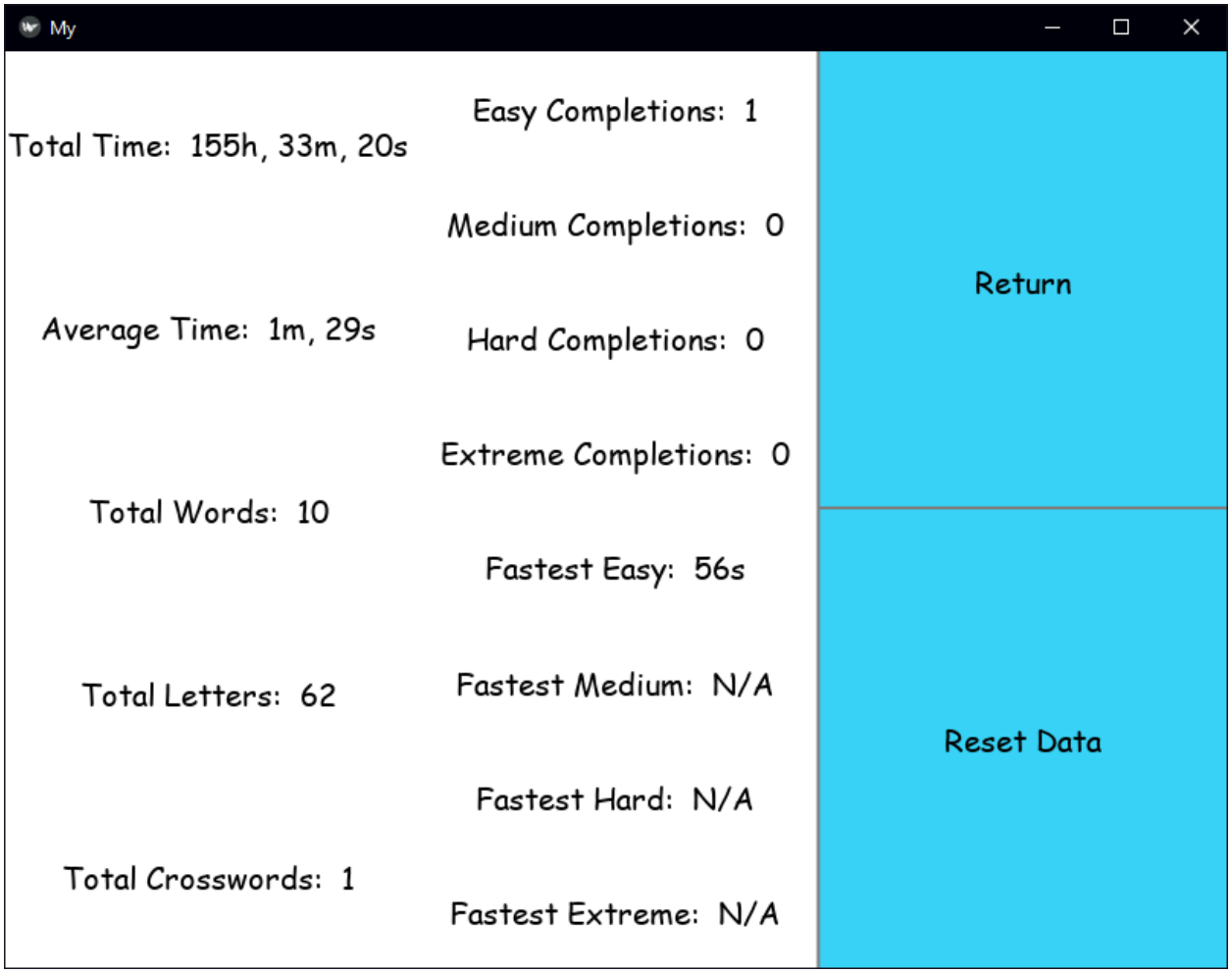
Test ID 20

This is to ensure the conversion algorithm functions in extreme cases.

This was tested on values which should be in minutes and hours too to ensure validation



Expected results:
Total Time: 155h, 33m, 20s
Average Time: 1m, 29s
Fastest Easy: 56s



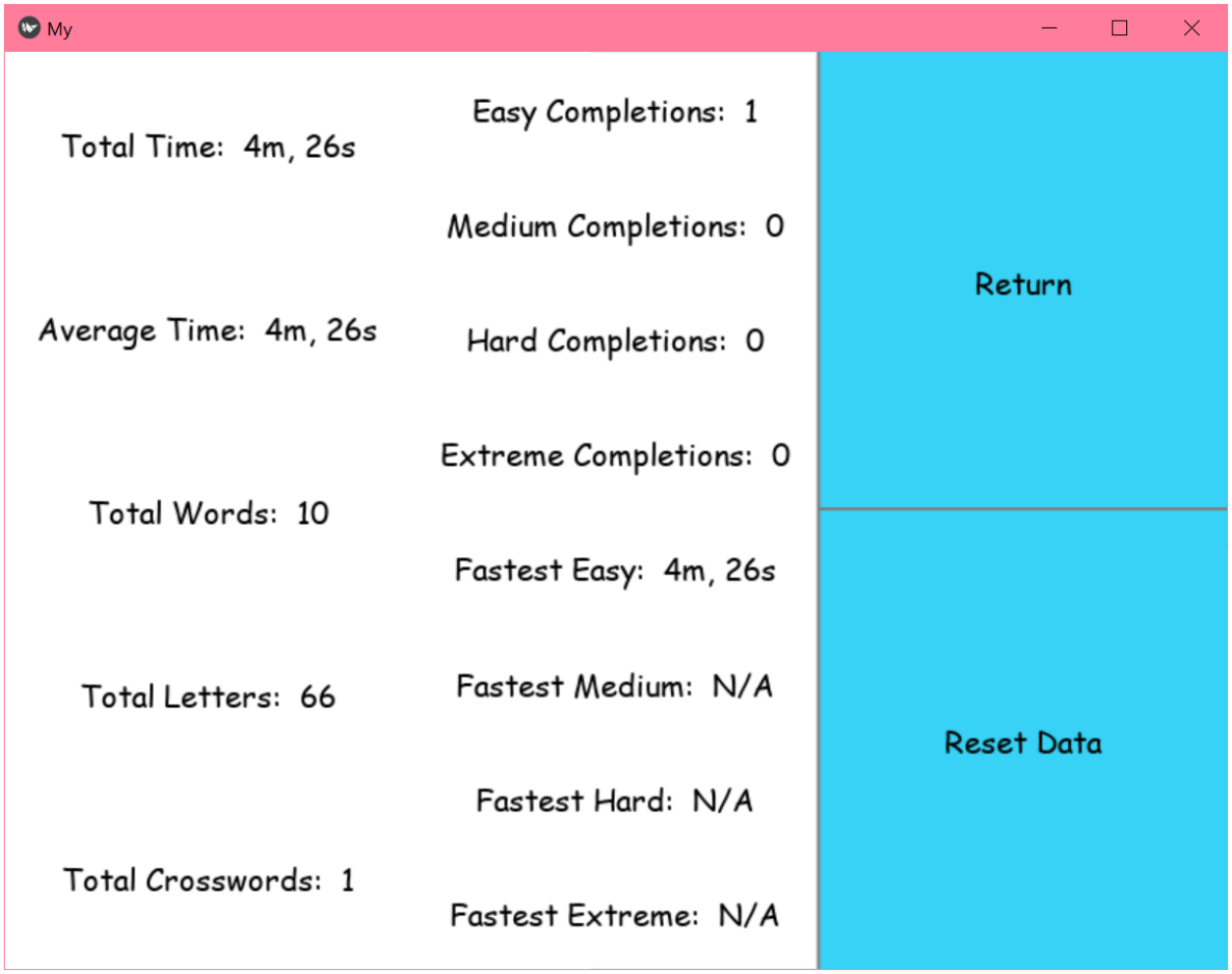
Results outputted as expected

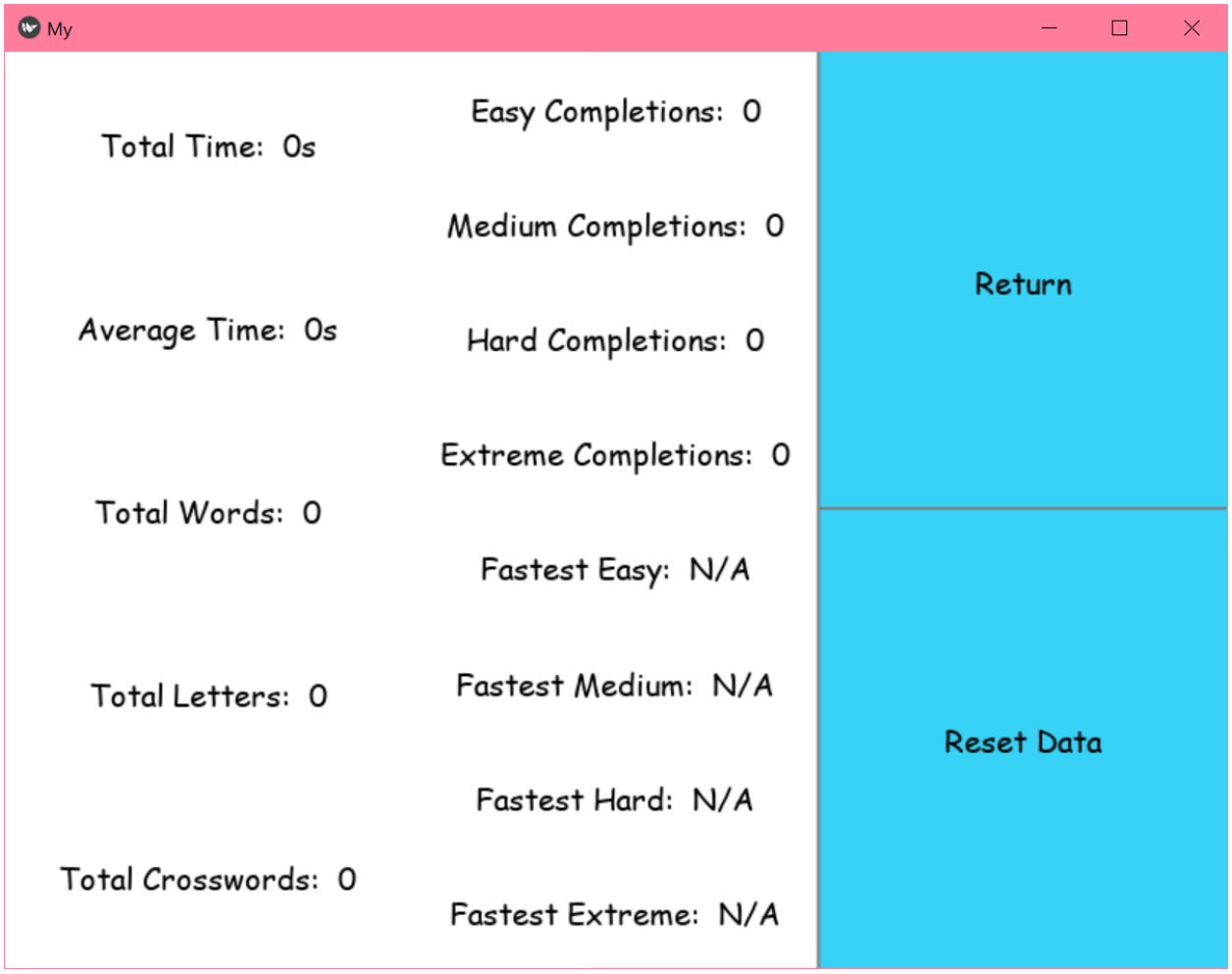
Reset Data

Also on the records page is the reset data button whose purpose is to do just that. My approach to this problem was to reuse the first_time_setup function to replace the file.

```
def reset_data(self):
    first_time_setup()
```

The following test starts with a file with unique data in it. After pressing the Reset Data button it is expected to display the default file when refreshed.





The data reverted to the default file, indicating a successful test.

Verify

The objective of this algorithm is to check if the user's inputs are all correct and if so displays the victory screen.

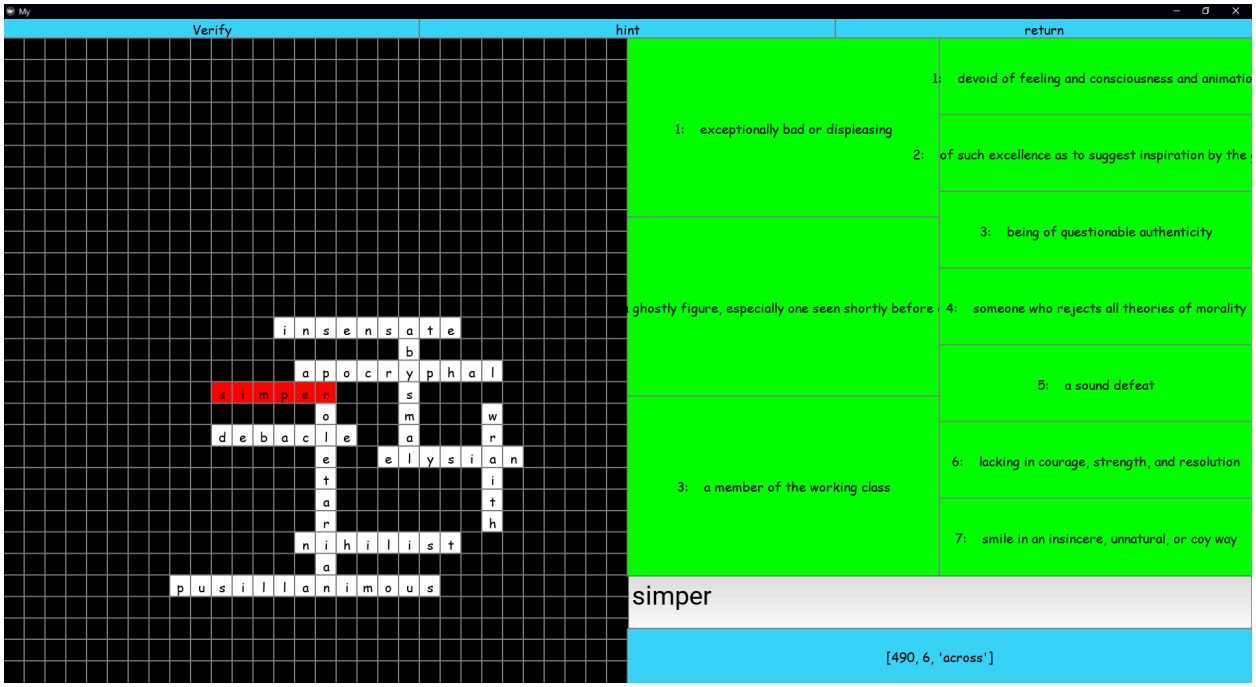
```
def verify(self):
    success = True
    for i in range(len(directionsdown)):
        length = directionsdown[i][1]
        pos = directionsdown[i][0]
        print(pos, length)

        for j in range(int(length)):
            print(self.my_buttons[int(pos) + 30 * j].text, ": ", crossword[(int(pos)) // 30 + j][(int(pos)) % 30])

            if self.my_buttons[int(pos) + 30 * j].text != crossword[(int(pos)) // 30 + j][((int(pos)) % 30)]:
                success = False

    for i in range(len(directionsacross)):
        length = directionsacross[i][1]
        pos = directionsacross[i][0]
        print(pos, length)

        for j in range(int(length)):
            print(self.my_buttons[int(pos) + j].text, ": ", crossword[(int(pos)) // 30][(int(pos)) % 30 + j])
            if self.my_buttons[int(pos) + j].text != crossword[(int(pos)) // 30][(int(pos)) % 30 + j]:
                success = False

    if success:
        print("GAMER!!!!!!!!!!!!")  # stattrack and winning screen
        self.my_buttons.color = 0, 1, 1, 1
        App.get_running_app().root.current = "victory"

    print(directionsdown)
    print(directionsacross)
```

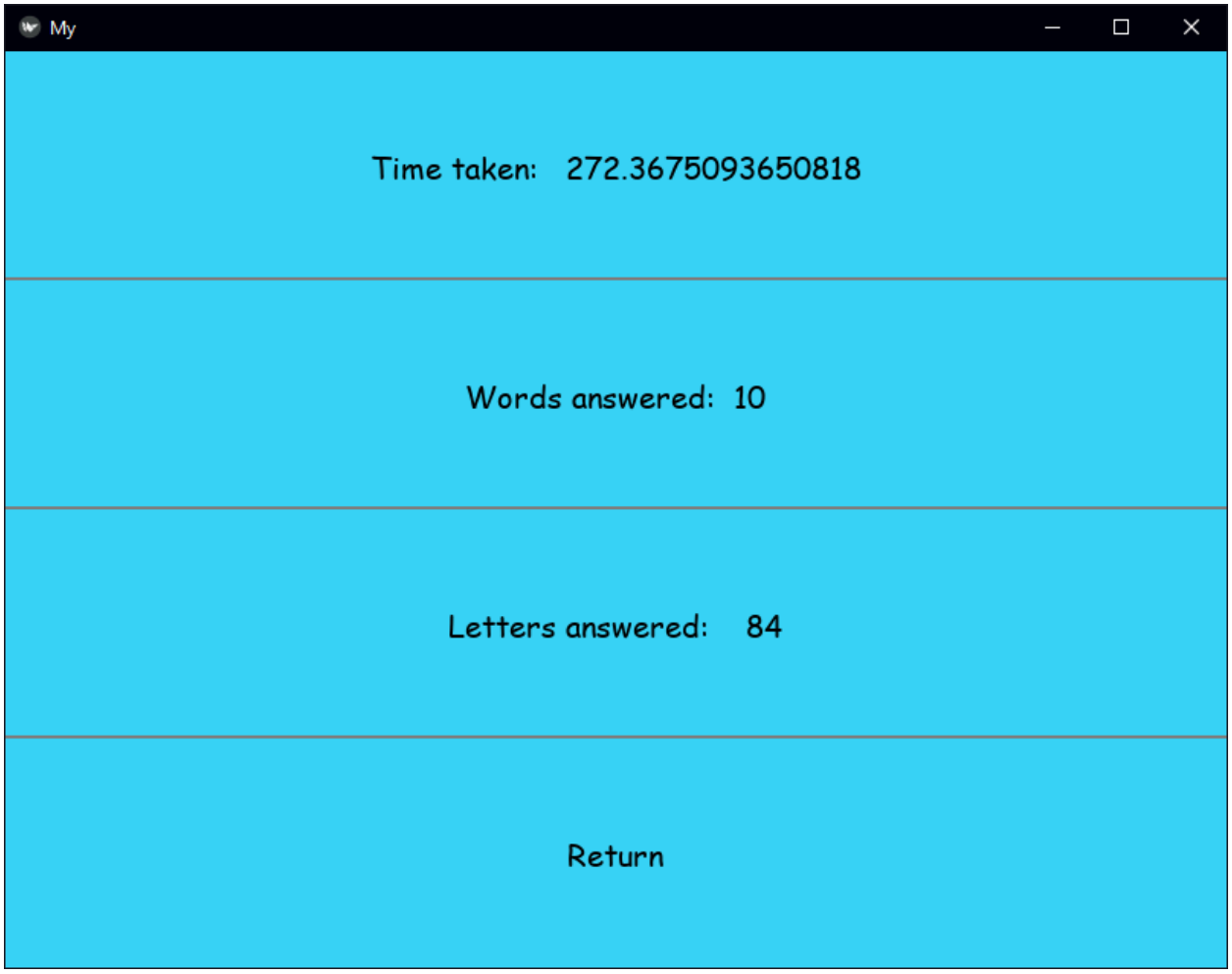
The program knows the direction of a word, its start position and its length. By using these it is able to map each letter correctly to its position on the grid. The positions have to be converted into the correct format to match the widgets though. For words going down, the next letter in the word is at original position + 30*j because the grid is a 30 by 30 and the widgets are stored from 0 to 900. For words going across every increasing letter is at the original position + j.

Any time these do not match the program flags success as False meaning that the end clause will be different. If success is True by the end the program navigates the user to the Victory Page, indicated by App.get_running_app().root.current = "victory"

Test ID 06

Verify pressed on known correct answers. Expected to navigate to victory page

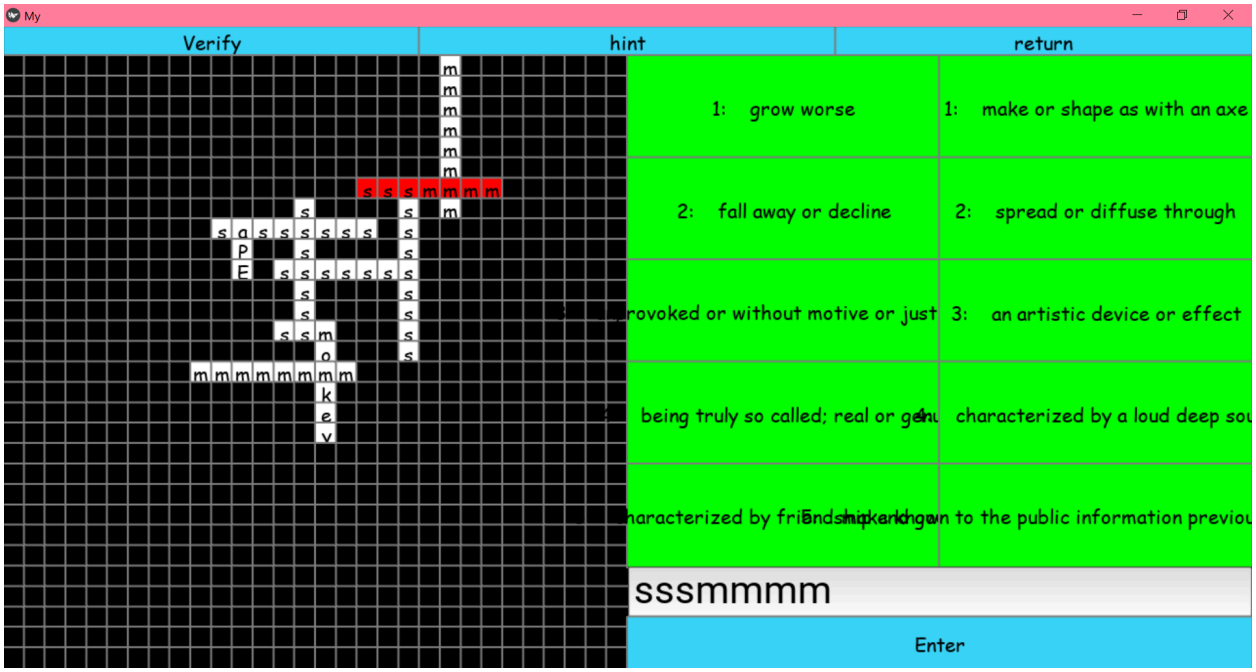


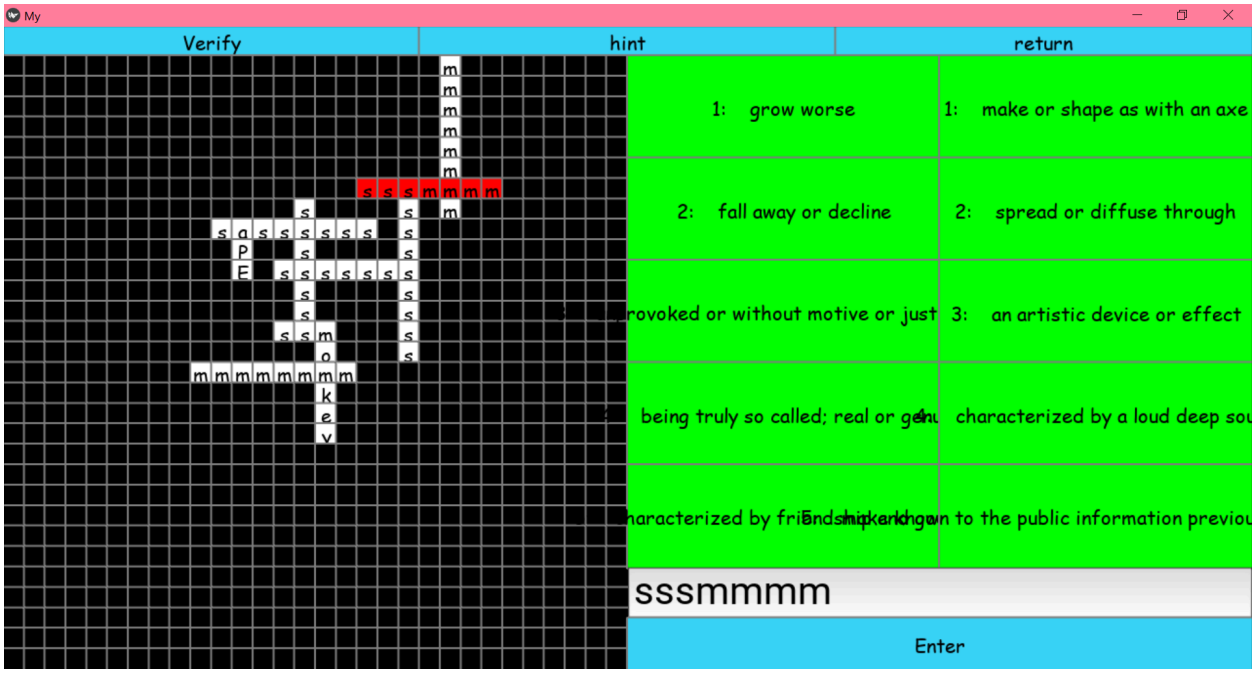


Verify pressed and user is successfully navigated to the victory page

Test ID 07

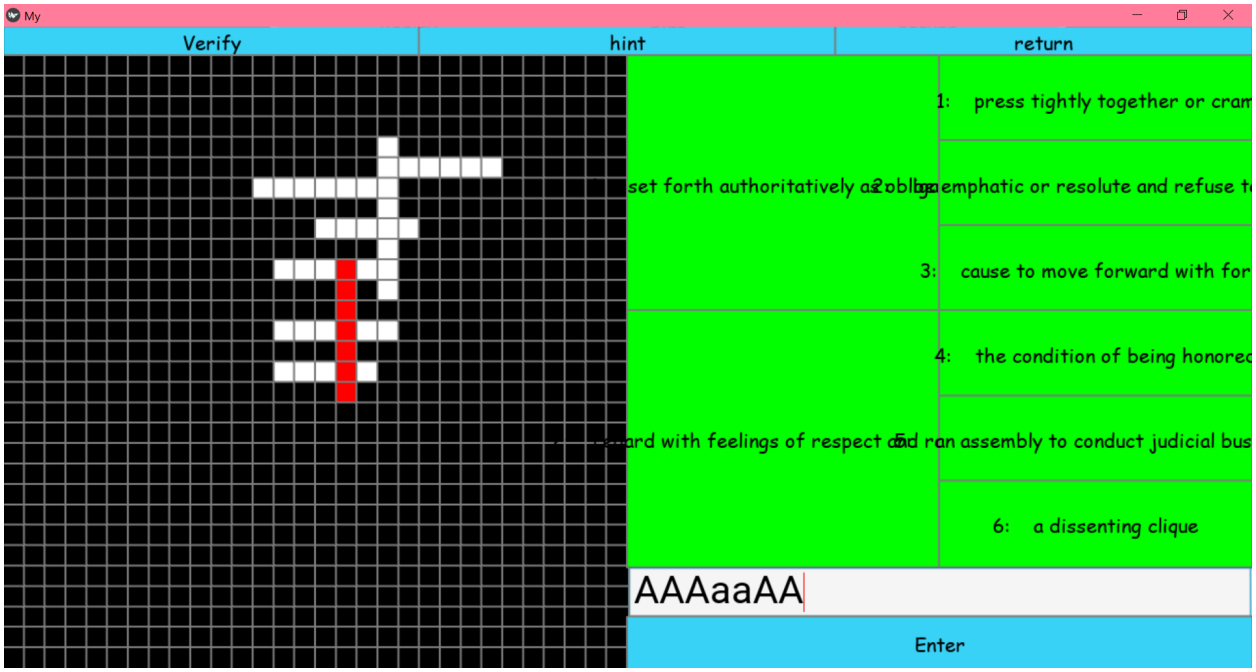
All fields filled but not correctly. Expected to fail

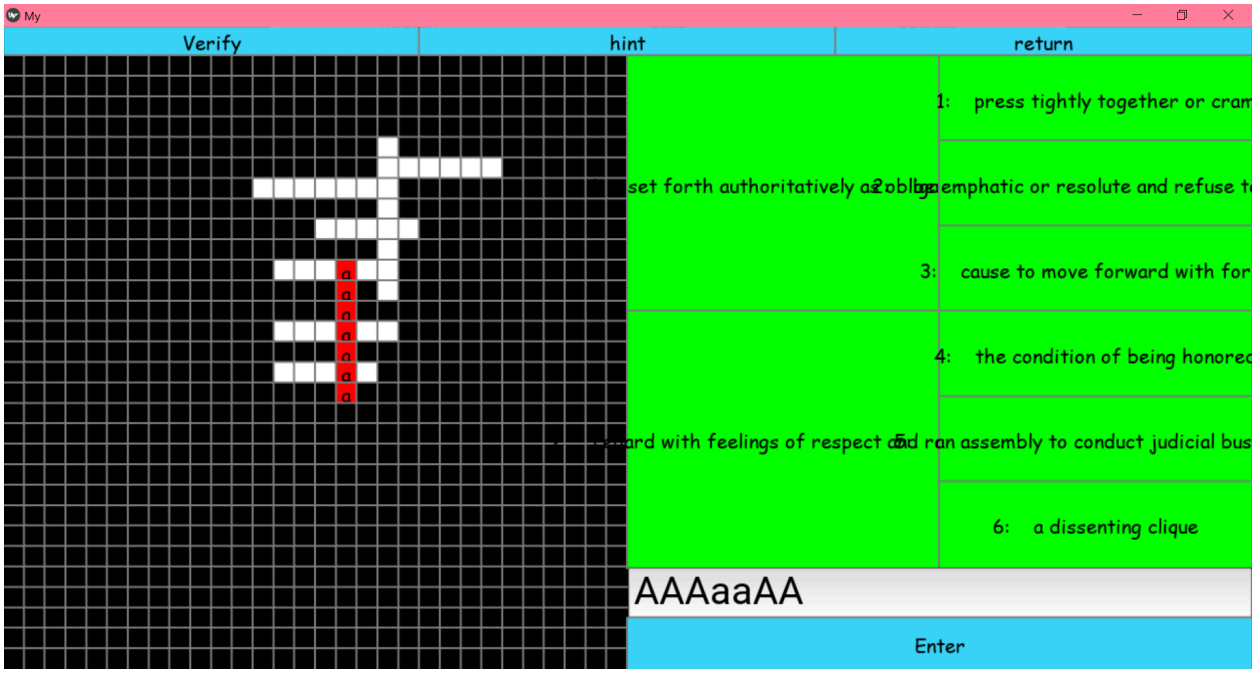




Does not verify, successful. While performing this test I also realised that the grid accepts both upper and lowercase characters. This went under the radar because the verify algorithm converts to lowercase when comparing and would still perform upon mixed case inputs.

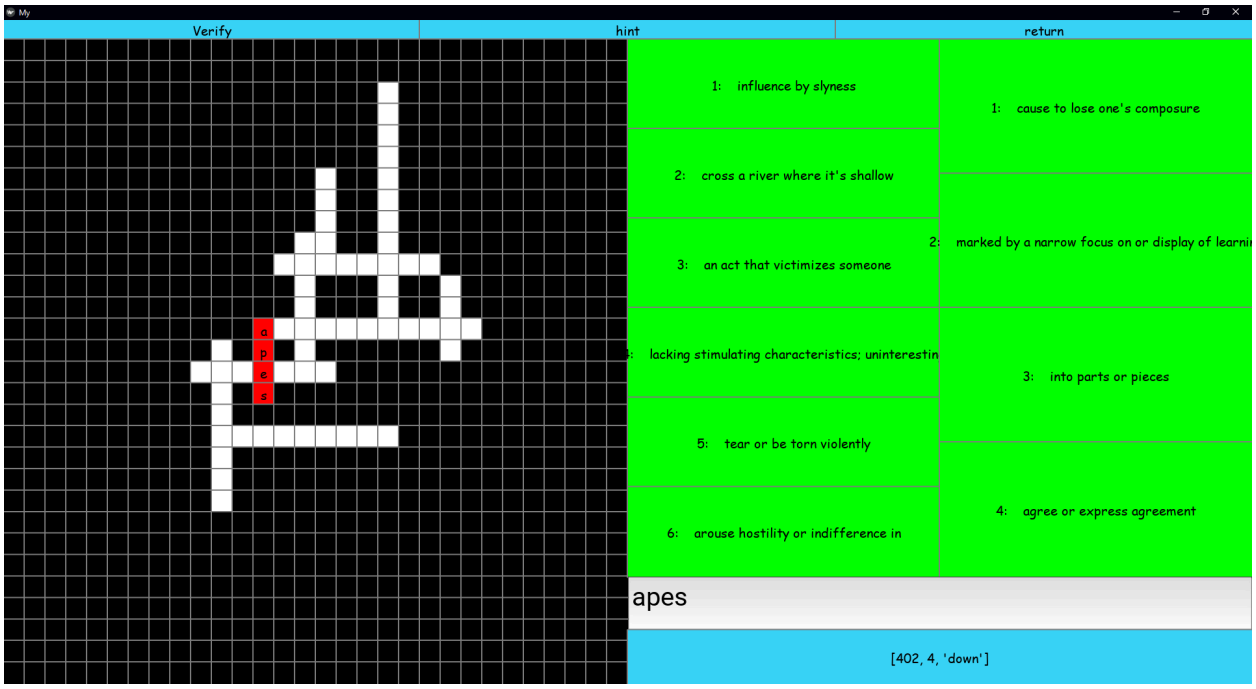
I fixed this by adding .lower() to the end of each inputted word and the program now runs as follows


My
Verify
hint
return
1: press tightly together or cram
set forth authoritatively as
2 oblig
emphatic or resolute and refuse t
3: cause to move forward with for
4: the condition of being honored
ard with feelings of respect and
5 an assembly to conduct judicial bus
6: a dissenting clique
AAAaaAA
Enter


My
Verify
hint
return
1: press tightly together or cram
set forth authoritatively as
2 oblig
emphatic or resolute and refuse t
3: cause to move forward with for
4: the condition of being honored
ard with feelings of respect and
5 an assembly to conduct judicial bus
6: a dissenting clique
AAAaaAA
Enter

Test ID 08

Incomplete grid with empty space. Upon verify press, expected nothing to happen



Does not navigate to victory page. Successful test

Give Hint

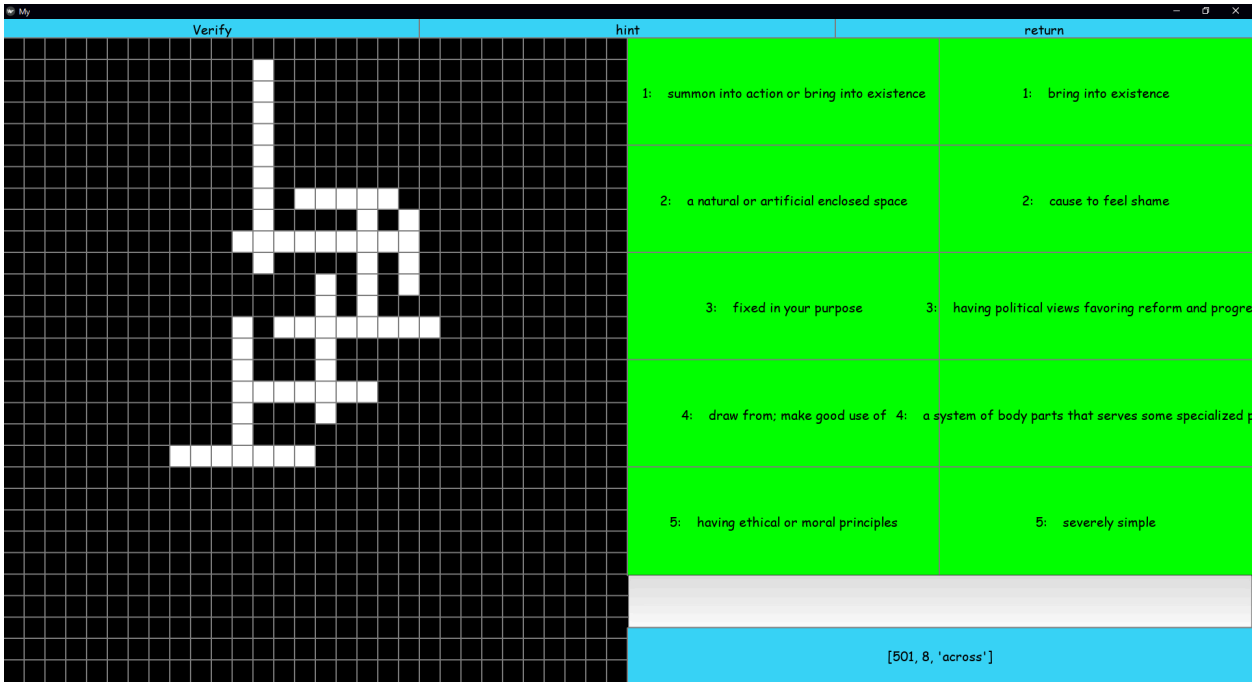
This algorithm's objective is to randomly fill in one letter on the grid with its correct letter.

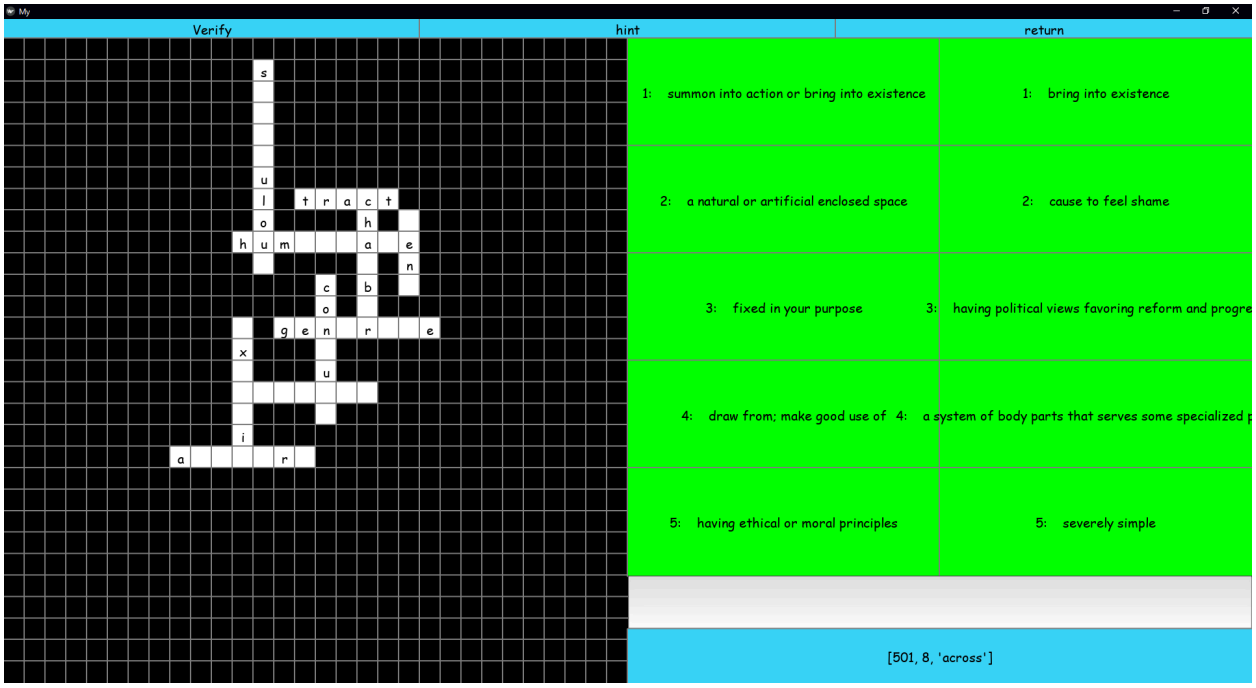
```
def givehint(self):
    if random.randint(0, 1) == 0:

        i = random.randint(0, len(directionsdown) - 1)
        length = directionsdown[i][1]
        pos = directionsdown[i][0]
        print(pos, length)

        j = random.randint(0, int(length))

        self.my_buttons[int(pos) + 30 * j].text = crossword[(int(pos)) // 30 + j][((int(pos)) % 30)]

    else:
        i = random.randint(0, len(directionsacross) - 1)
        length = directionsacross[i][1]
        pos = directionsacross[i][0]
        print(pos, length)

        j = random.randint(0, int(length))

        self.my_buttons[int(pos) + j].text = crossword[(int(pos)) // 30][(int(pos)) % 30 + j]
```

It works by randomly selecting a direction, a word from that direction and a letter from that word. The rest of the program reuses some of the verification function in mapping back to the grid.

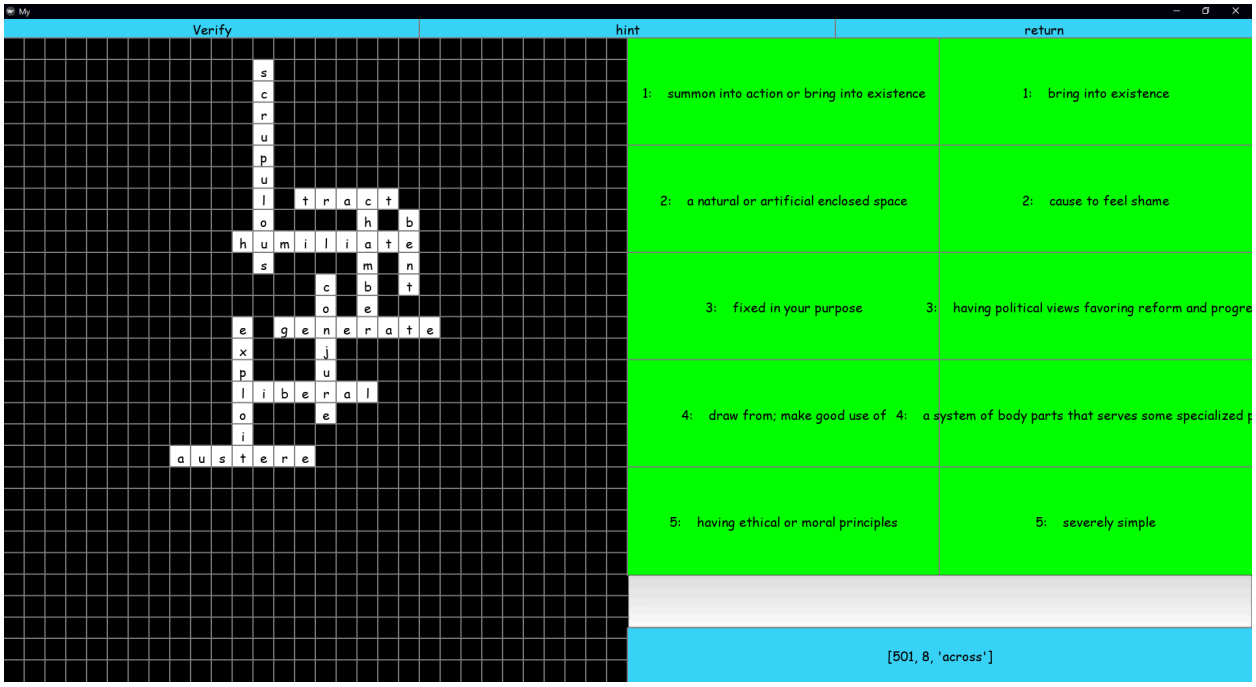
Test ID 09

Upon hint presses, expected to fill in a random spots correctly.





Grid being correctly filled



^^ Testing evidence of the hint button functioning properly to complete an entire crossword. Initial grid, a few hints given and the entire completed grid done by hints.

Colour Changes

My application requires dynamic colour changes in order to primarily facilitate skins, accessibility options and the highlight function. As stated in the key variables I decided on having an array hold all the colour values for a certain theme/skin and have the user be able to choose one of these.

The base procedure to change colours is this one

```
def colour_change(self, *largs):
    print(largs[0])
    if largs[0] == "dark":
        self.palette = self.dark
    if largs[0] == "default":
        self.palette = self.default

    # colour_bg = palette[0]
    # colour_widget = palette[1]
    # colour_highlight = palette[2]
    # colour_grid = palette[3]
    # colour_text = palette[4]
```

This iteration only has 2 options to pick but even so proves functional.

```
default = ListProperty([(1, 1, 1, 1), (0, 1, 0, 1), (1, 0, 0, 1), (1, 1, 1, 1), (0, 0, 0, 1)])  # default, constant
dark = ListProperty([(1, 1, 1, 1), (1, 1, 1, 1), (1, 1, 1, 1), (1, 1, 1, 1), (1, 1, 1, 1)])  # dark mode, constant
palette = ListProperty([(1, 1, 1, 1), (0, 1, 0, 1), (1, 0, 0, 1), (1, 1, 1, 1), (0, 0, 0, 1)])  # colour variable
```

The arrays themselves are stored within the Settings class as that is where they are referenced the most.

Based on the format of Kivy and how layouts work and have referenceable children instead of changing each widget one by one I would reference the children of a parent widget inside of for loops instead. This was perfect for my needs as I had already structured the layouts in a distinguished and logical way.

```
def colour_update(self):
    print(self.manager.get_screen("home").ids.settingsbtn)
    for i in self.manager.get_screen("home").ids.homefloat.children:
        i.background_color = self.palette[0]  # bg
        i.color = self.palette[4]             # text
    for i in self.manager.get_screen("difficulty").ids.difficultyfloat.children:
        i.background_color = self.palette[1]  # widget
        i.color = self.palette[4]             # text
    for i in self.manager.get_screen("cword").ids.cwordbox.children:
        i.background_color = self.palette[1]  # widget
        i.color = self.palette[4]             # text
    self.manager.get_screen("cword").ids.cwordenter.background_color = self.palette[1]
    for i in self.manager.get_screen("record").ids.recordsbox.children:
        i.background_color = self.palette[1]  # widget
        i.color = self.palette[4]             # text
    for i in self.ids.settingsbox1.children:
        i.background_color = self.palette[1]  # widget
    for i in self.ids.settingsbox2.children:
        i.background_color = self.palette[1]  # widget
    Window.clearcolor = self.palette[0]       # bg
```

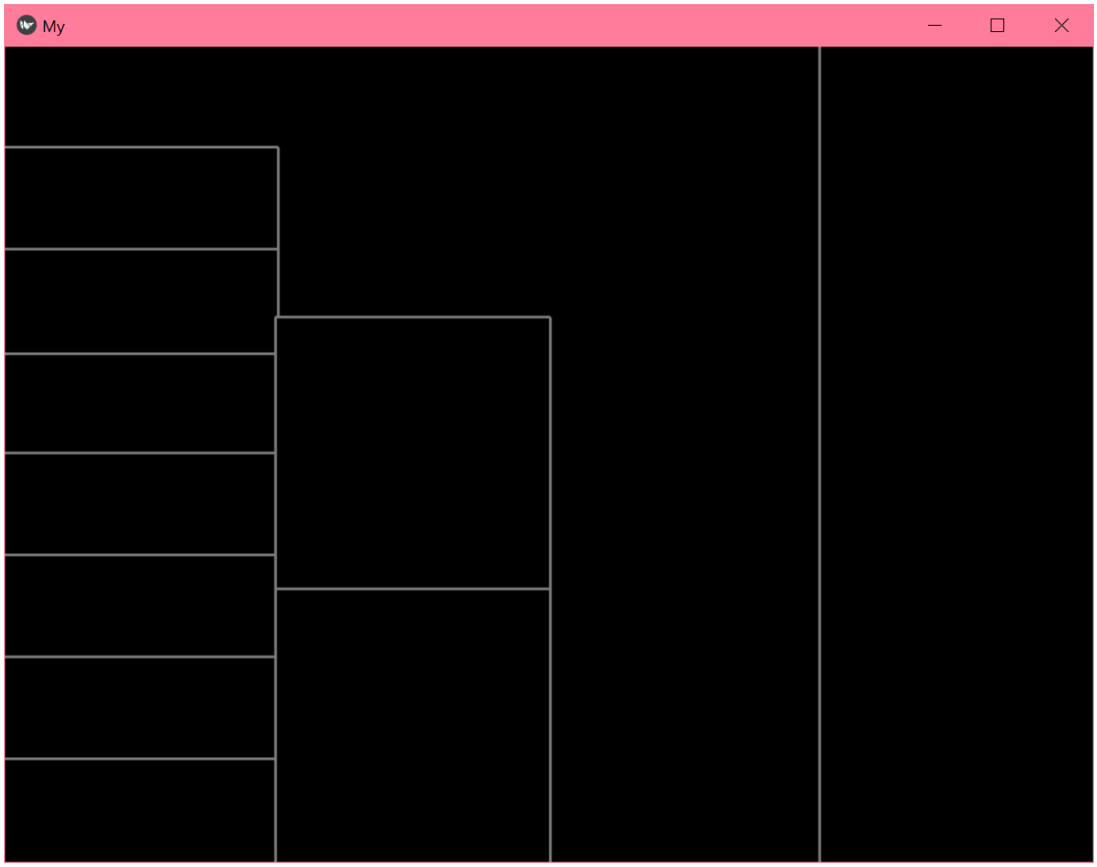
As you can see above this function runs through all parent widgets of the interface and updates their assigned background and text colours. Comments are made to make it easier to see what is being changed in each line while allowing it to all be done within a single array. The Window.clearcolor at the bottom changes the main background for every screen.

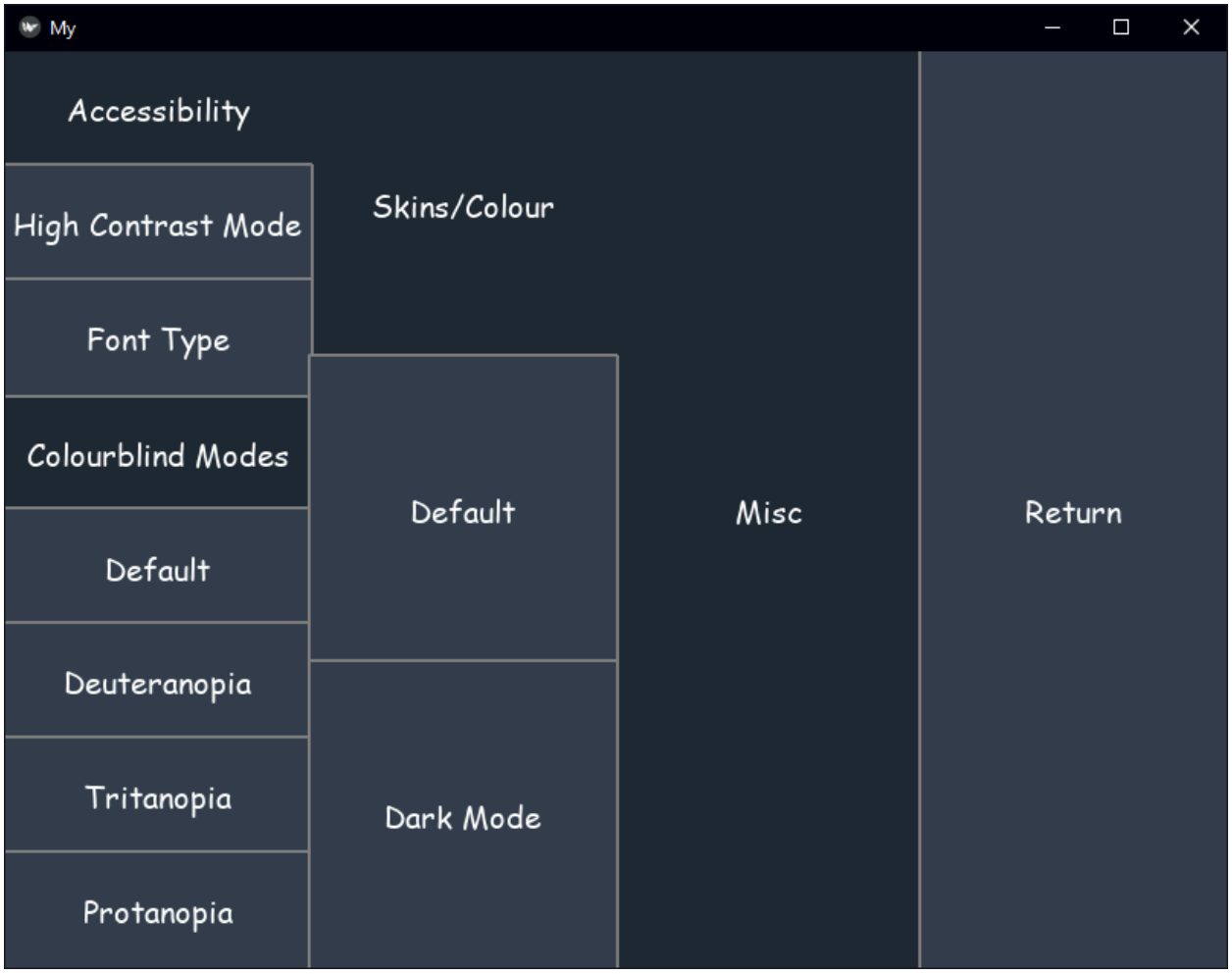
The main problem with designing this algorithm was passing the data through between screens but after enough testing and research I decided to declare the variables within the class and have it referenced as such.

Test ID 21

Upon Dark Mode button press, widgets are updated to that of dark mode


My
Accessibility
High Contrast Mode
Font Type
Colourblind Modes
Default
Deuteranopia
Tritanopia
Protanopia
Skins/Colour
Default
Dark Mode
Misc
Return


My

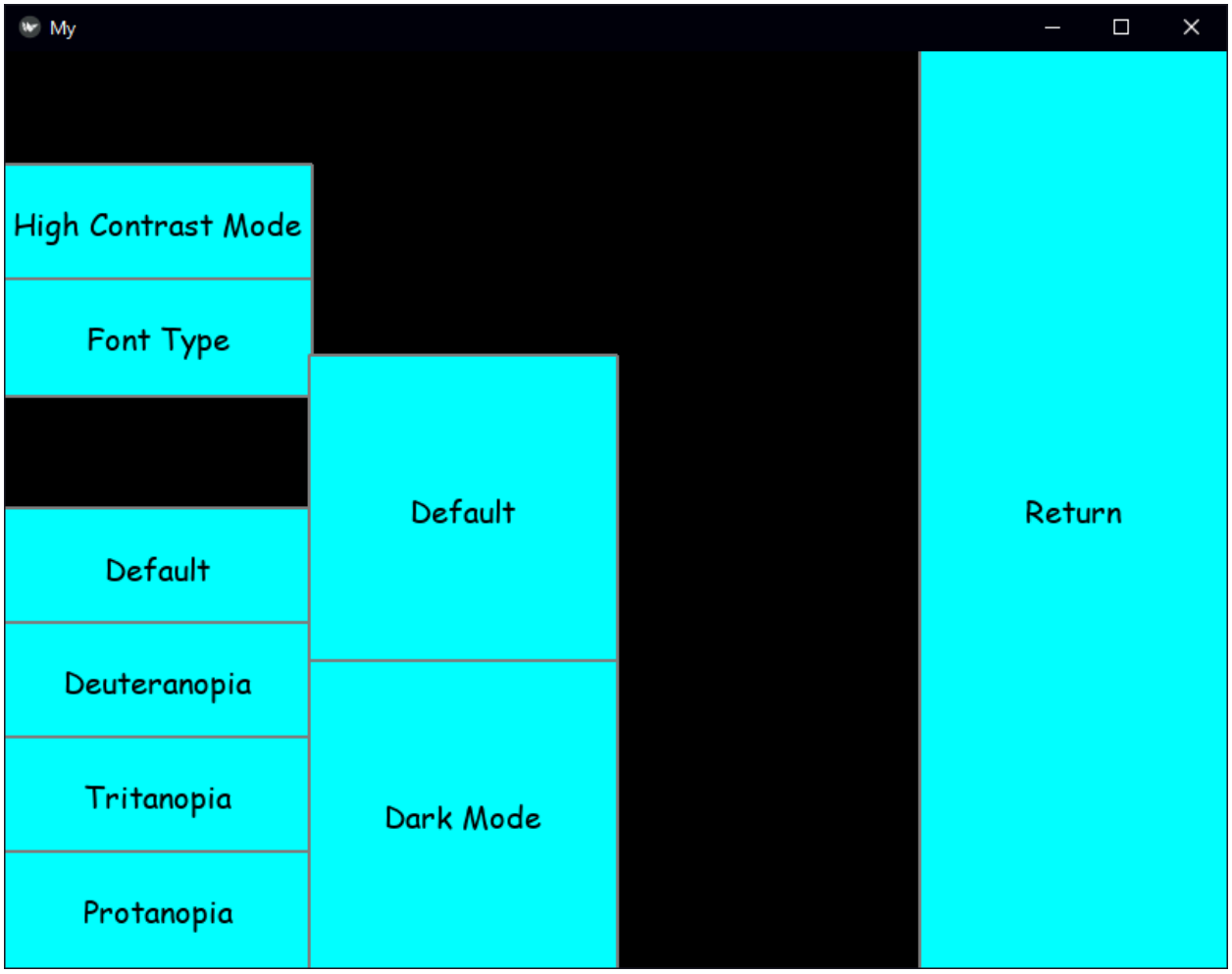


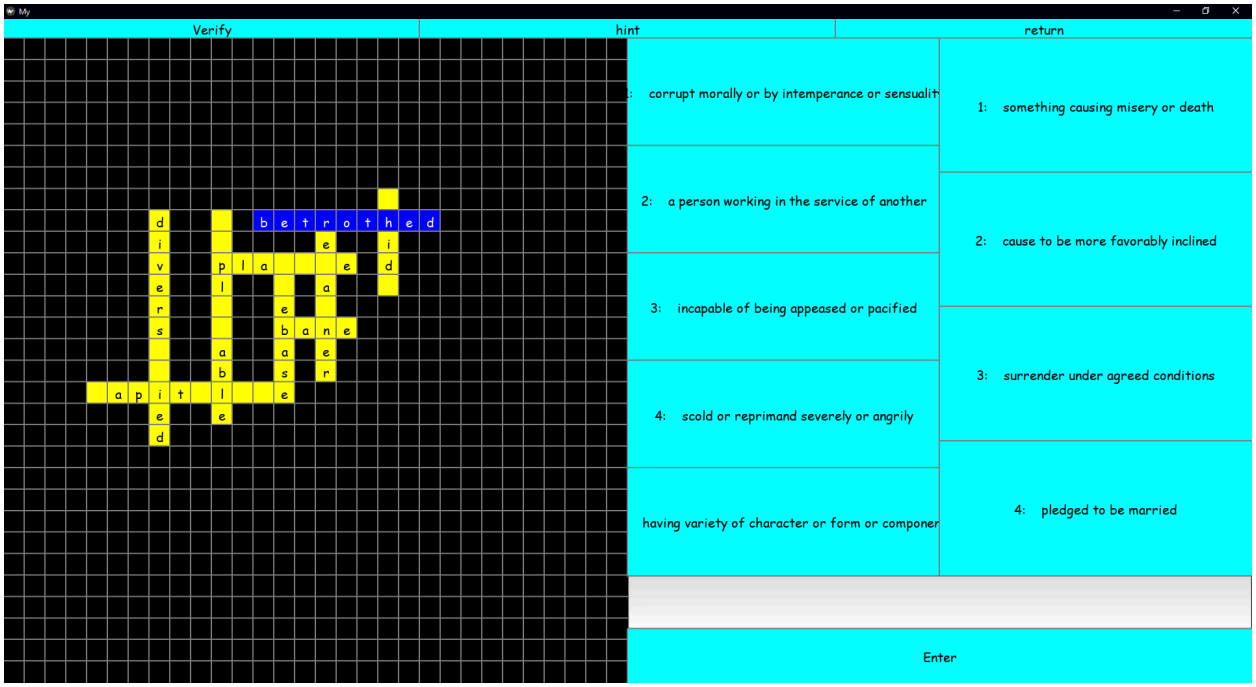
Widgets changed as expected, successful

High contrast mode matches that of online applications.

Test ID 22

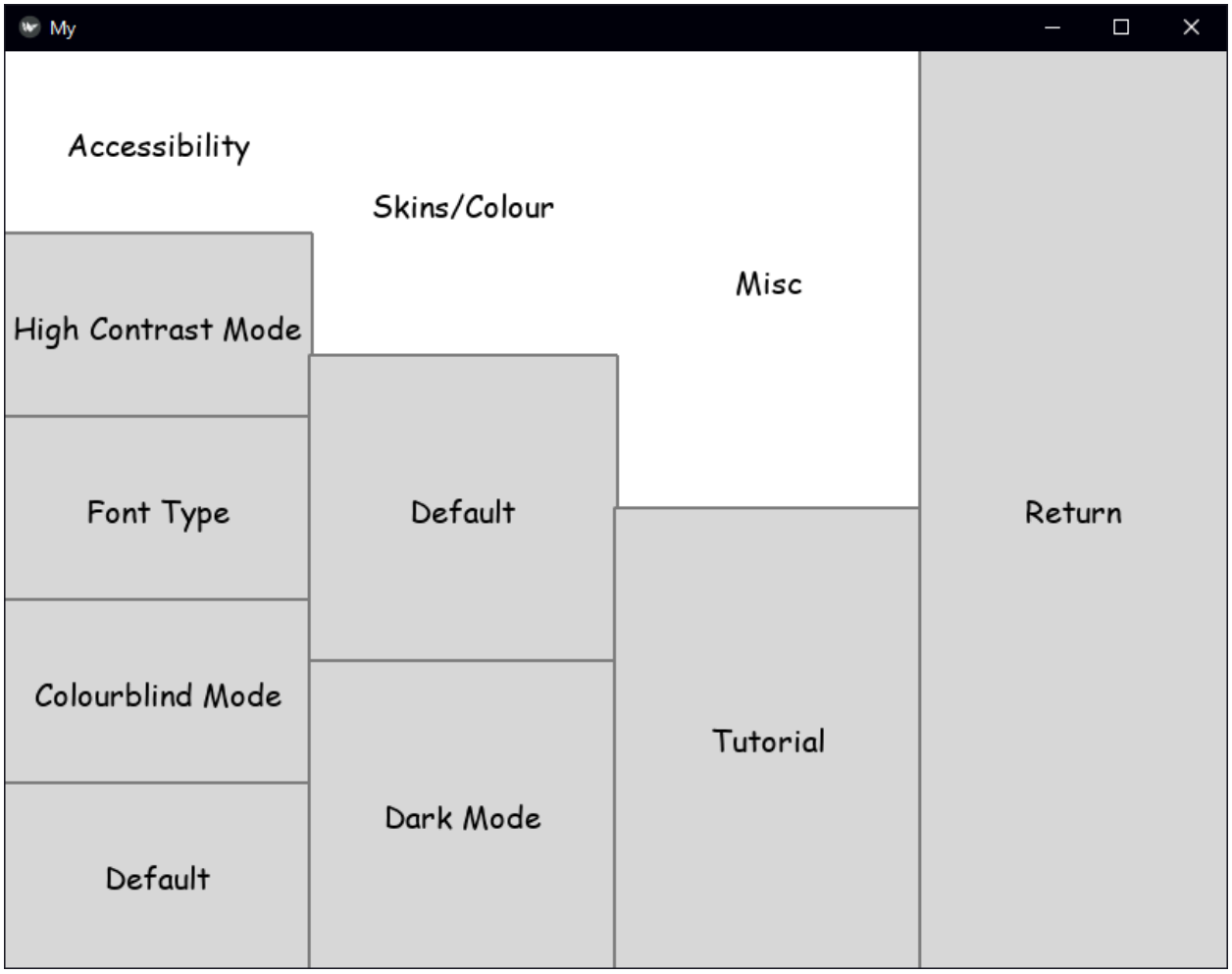
Expected high contrast:


My
High Contrast Mode
Font Type
Default
Deuteranopia
Tritanopia
Protanopia
Default
Dark Mode
Return


Verify
hint
return
corrupt morally or by intemperance or sensualit
2: a person working in the service of another
3: incapable of being appeased or pacified
4: scold or reprimand severely or angrily
having variety of character or form or componer
1: something causing misery or death
2: cause to be more favorably inclined
3: surrender under agreed conditions
4: pledged to be married
Enter

Test ID 23

Expected: displays default skin


My
Accessibility
Skins/Colour
Misc
High Contrast Mode
Font Type
Colourblind Mode
Default
Default
Dark Mode
Tutorial
Return

Expected outcome, proving skins do not overlap (only one active at once), success

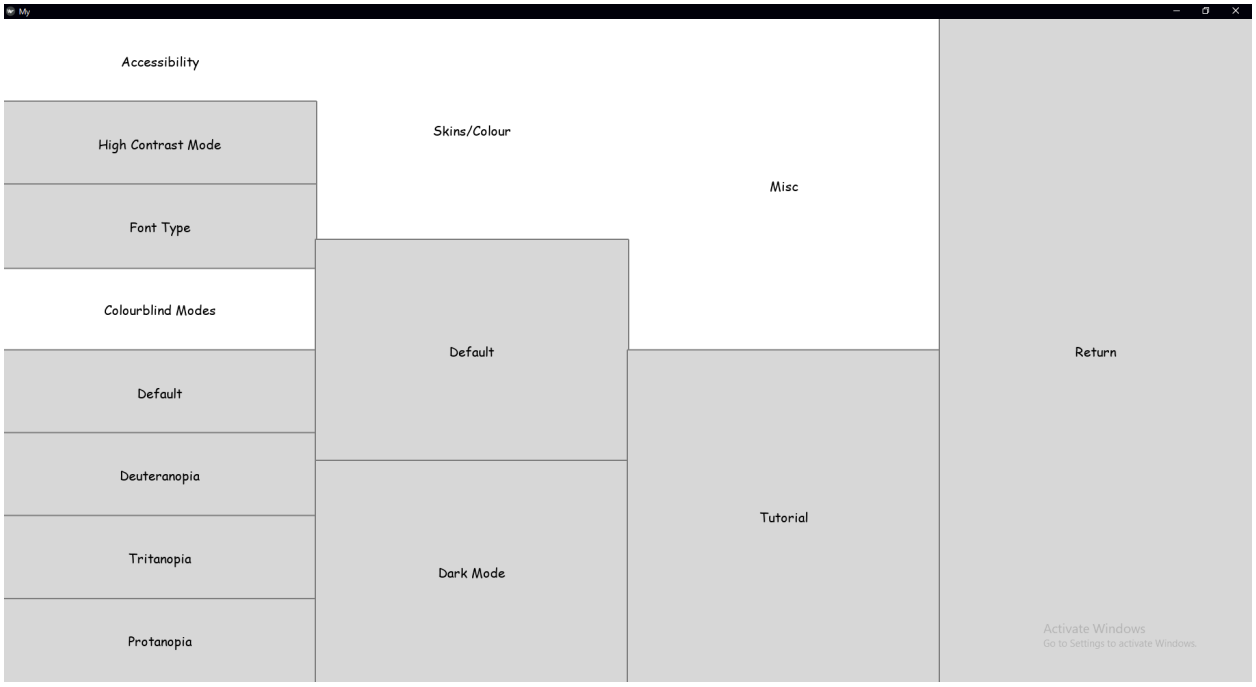
I did some research around colourblind standards to help design my features.

www.usability.gov states that “content areas should be monochromatic with the font color and background at the opposite ends of the color saturation poles (ie., black text on a white background).” With this in mind I decided to make one skin for all general colourblind variants which would consist of primarily black, white and grey stacked accordingly.

```
colourblind = ListProperty([(1,1,1,1),(.85,.85,.85,1),(.5,.5,0,1),(1,1,1,1),(0,0,0,1),(0,0,0,1)])
```

Test ID 24

Expected to change to colourblind skin upon press



Successfully changes as expected. The unnecessary widgets will be removed before the final iteration.

To change the font I needed a function to do so. Since the main reason the font was there was to switch to accessibility I decided to make the button switch between the comic sans (accessibility) and arial (classic). The following algorithm runs when the button is pressed

```
def font_change(self):
    if self.font == "comic":
        self.font = "arial"
    else:
        self.font = "comic"
```

To change the font for each widget I decided the easiest way to do so would be by implementing them inside the preexisting colour_update() function.

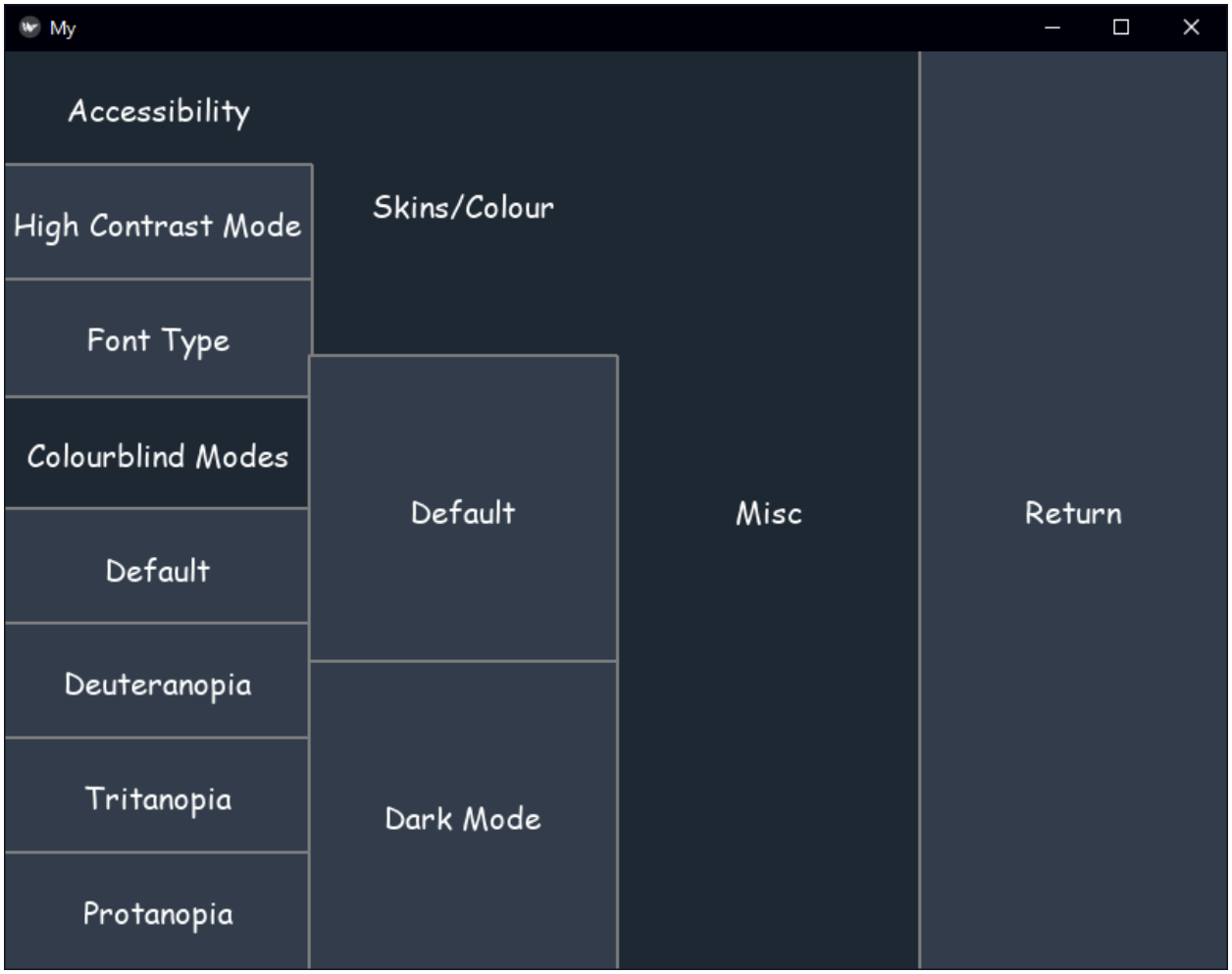
```
def colour_update(self):
    print(self.manager.get_screen("home").ids.settingsbtn)
    for i in self.manager.get_screen("home").ids.homefloat.children:
        i.background_color = self.palette[1]  # widget
        i.color = self.palette[4]             # text
        i.font_name = self.font               # font
    for i in self.manager.get_screen("difficulty").ids.difficultyfloat.children:
        i.background_color = self.palette[1]  # widget
        i.color = self.palette[4]             # text
        i.font_name = self.font
    for i in self.manager.get_screen("cword").ids.cwordbox.children:
        i.background_color = self.palette[1]  # widget
        i.color = self.palette[4]             # text
        i.font_name = self.font
    self.manager.get_screen("cword").ids.cwordenter.background_color = self.palette[1]
    for i in self.manager.get_screen("record").ids.recordsbox.children:
        i.background_color = self.palette[1]  # widget
        i.color = self.palette[4]             # text
        i.font_name = self.font
    for i in self.ids.settingsbox1.children:
        i.background_color = self.palette[1]  # widget
        i.color = self.palette[4]             # text
        i.font_name = self.font
    for i in self.ids.settingsbox2.children:
        i.background_color = self.palette[1]  # widget
        i.color = self.palette[4]             # text
        i.font_name = self.font
    self.ids.settingsreturn.background_color = self.palette[1]

    self.ids.settingsreturn.color = self.palette[4]
    self.ids.settingsreturn.font_name = self.font
    self.ids.misc.color = self.palette[4]
    self.ids.misc.font_name = self.font
    Window.clearcolor = self.palette[0]       # bg
```

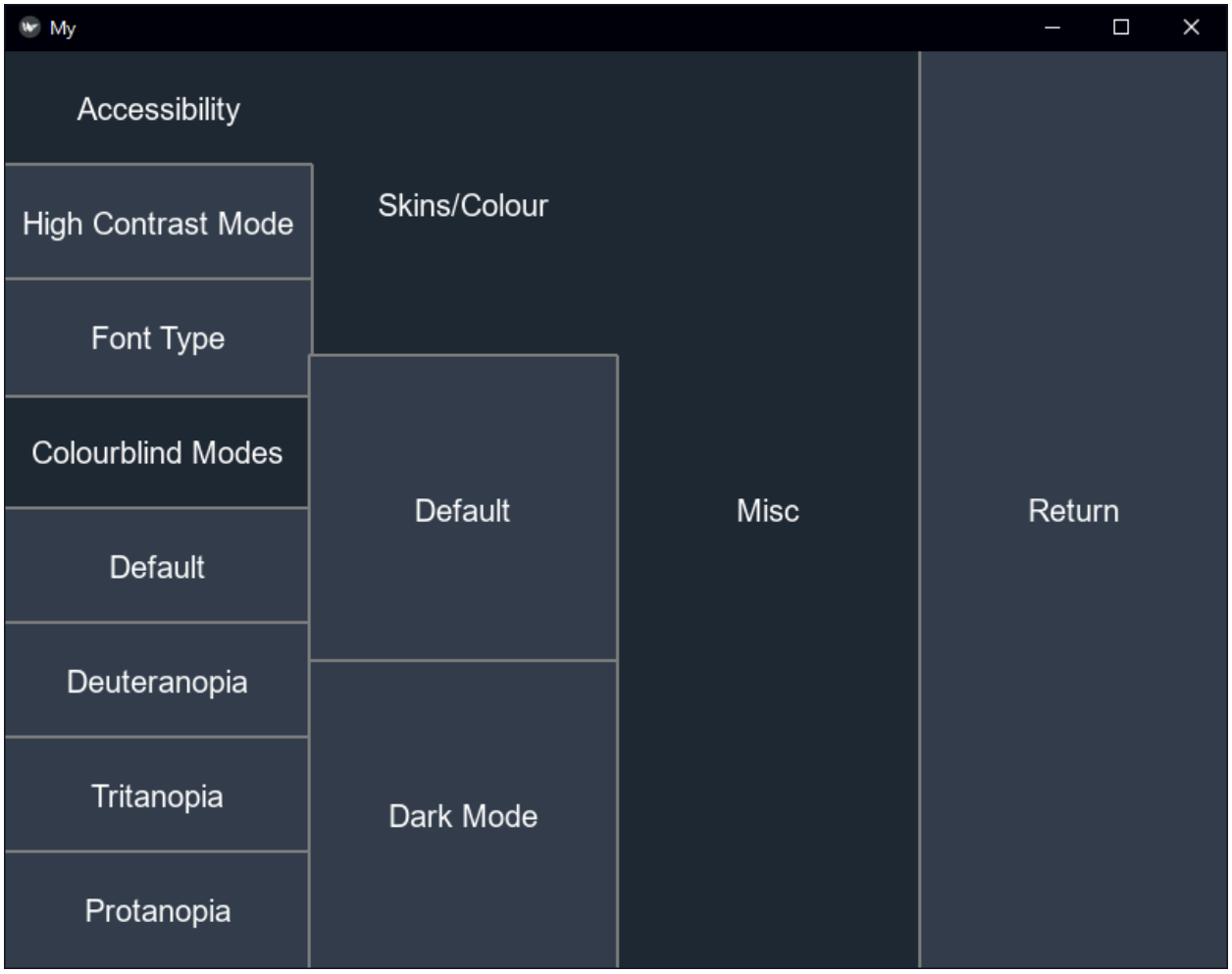
When the “Font Type” button is pressed these are both run one after the other

```
Button:
    text:"Font Type"
    on_release:
        root.font_change()
        root.colour_update()
```

Test ID 28

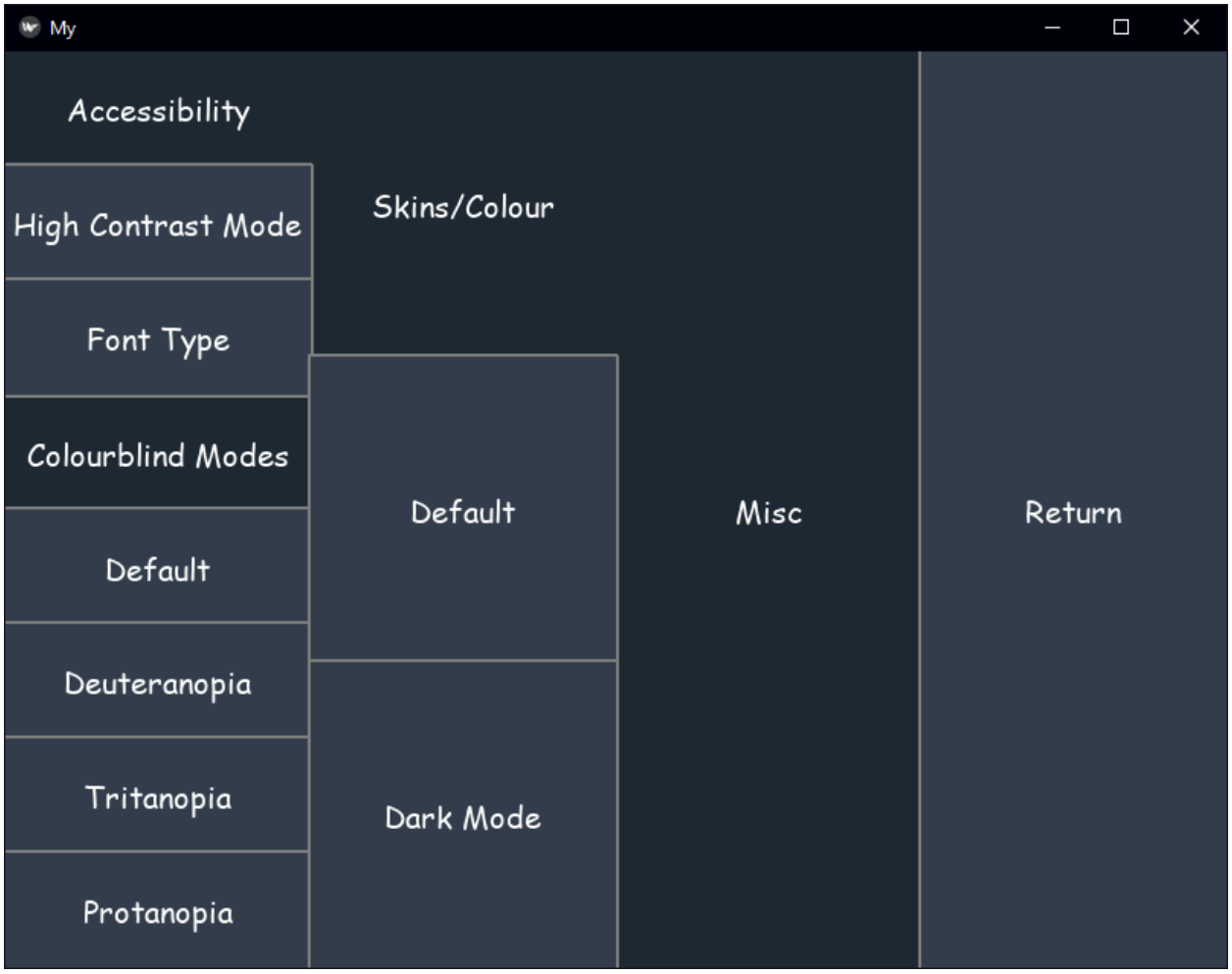
Expected font to go to arial upon press when starting on comic sans:





Expected result, success

If pressed again expected to return to comic sans:



As expected, success

Highlight

This is the function for the crossword page intended to highlight the word the user has selected on the crossword grid for increased visibility and clarity. Initially I could easily change the colour of the words's positions selected but struggled with changing them back after unselected.

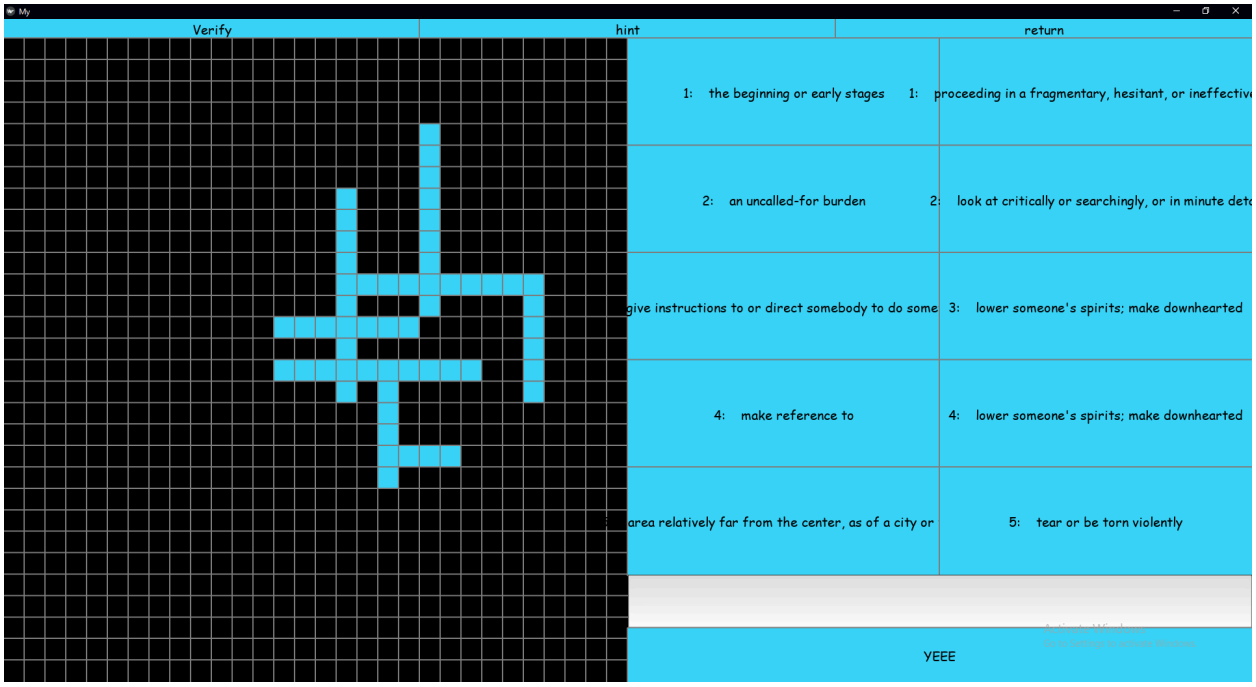
```
def highlight(self, *largs):

    if largs[0][2] == "down":
        for l in range(largs[0][1]):
            self.my_buttons[largs[0][0]+30*l].background_color = 1,0,1,1

    if largs[0][2] == "across":
        for l in range(largs[0][1]):
            self.my_buttons[largs[0][0] + l].background_color = 1, 0, 1, 1

    self.text_input.text = ""
```

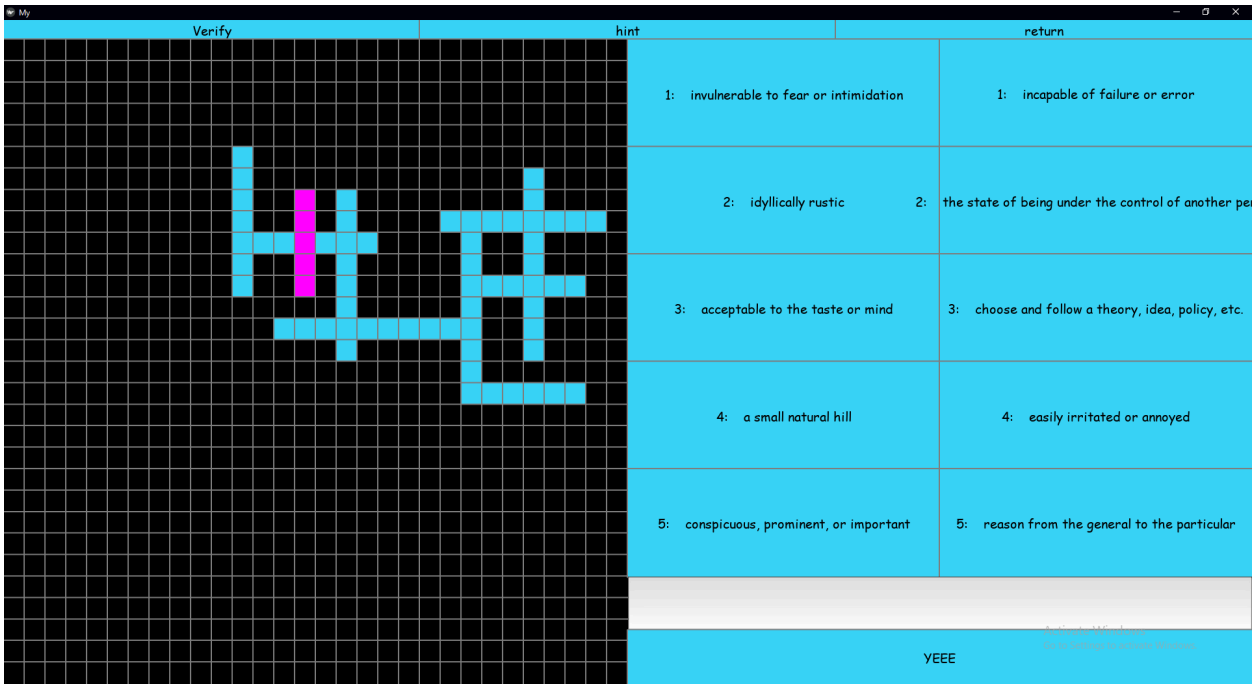
This code works my taking in the position data through whichever button was pressed, determined here upon creation

```
btn.bind(on_press=partial(self.highlight,directionsdown[k]))
```

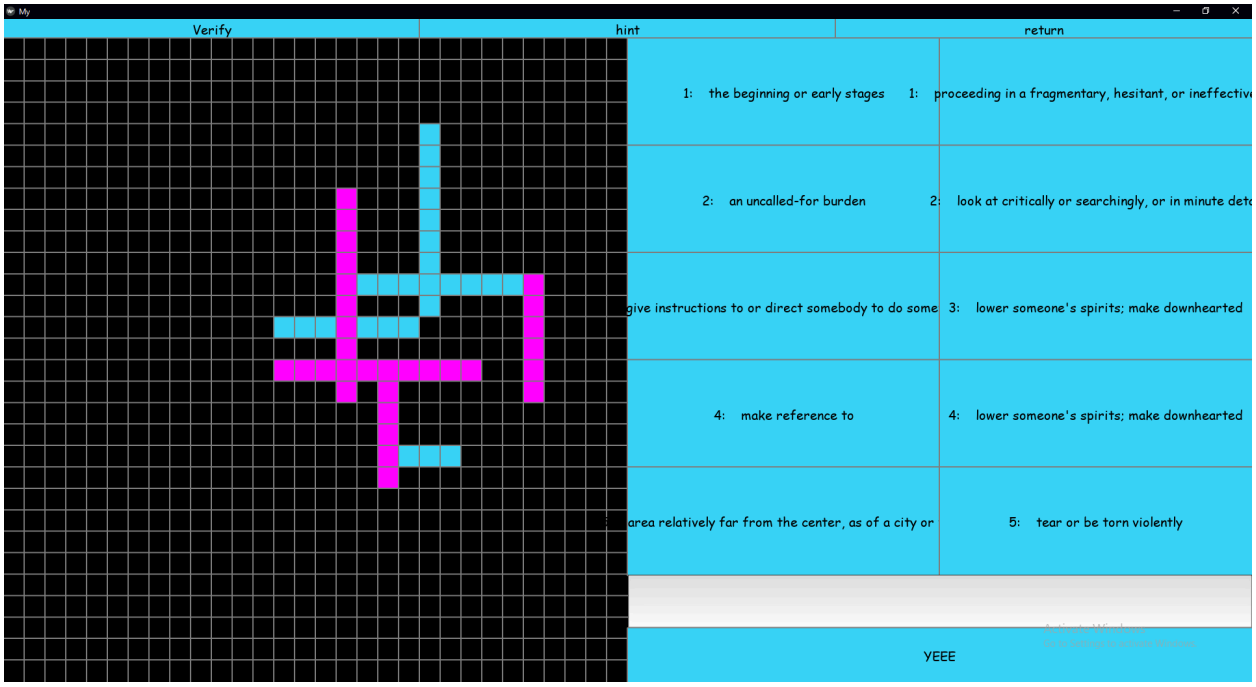
Test ID 24

Check that the highlight function works





The code works fine in highlighting, but upon selecting another clue:



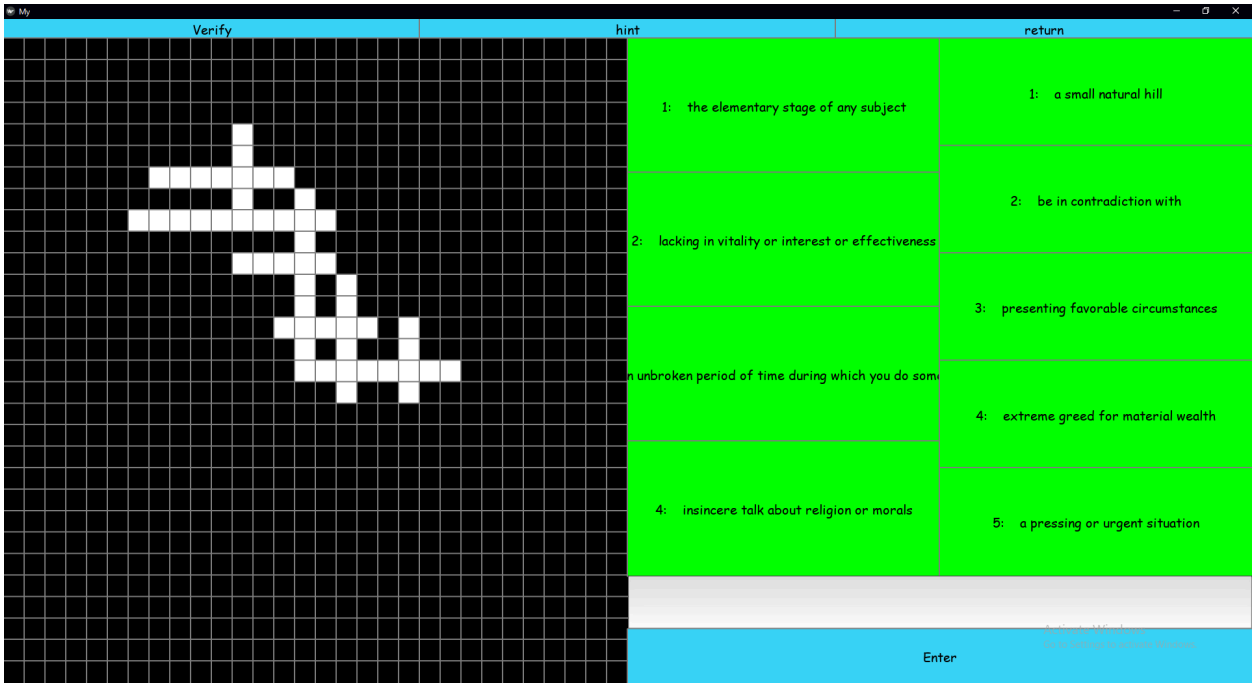
The old one doesn't lose its highlight; unsuccessful

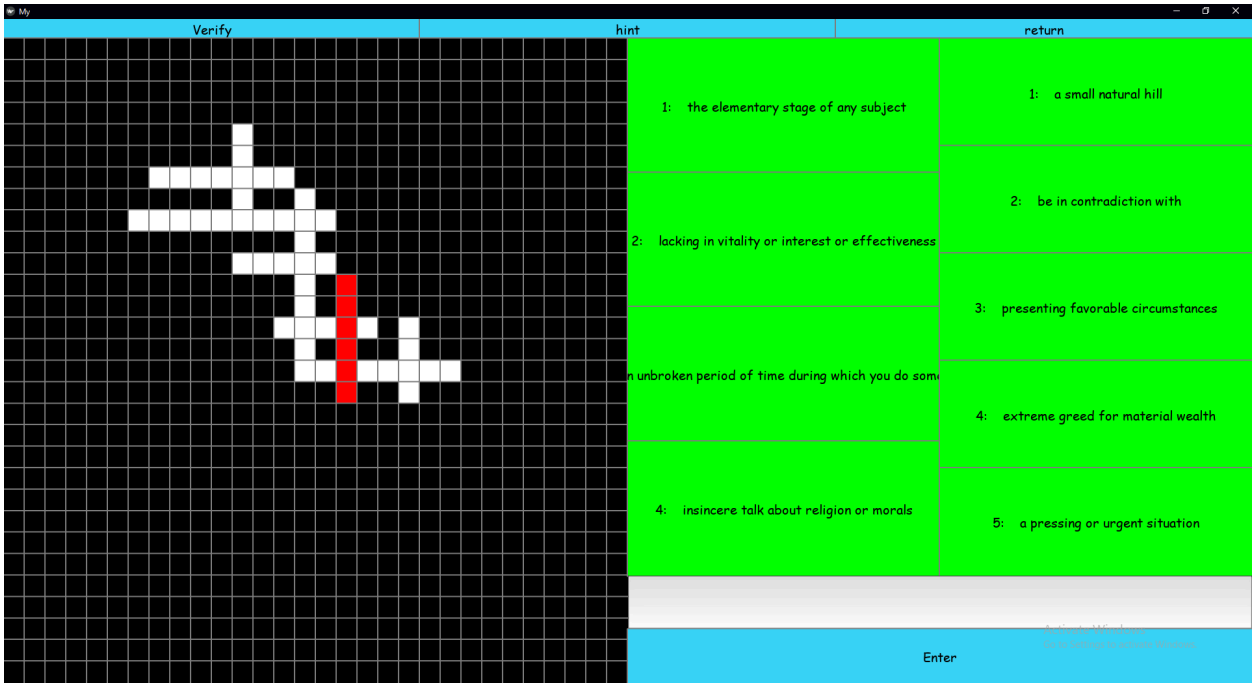
My fix for this was to have the position value of the last updated clue stored in a variable called selected and before the next one is highlighted the old one is unhighlighted. After the new one is highlighted it takes the place of the old one in selected:

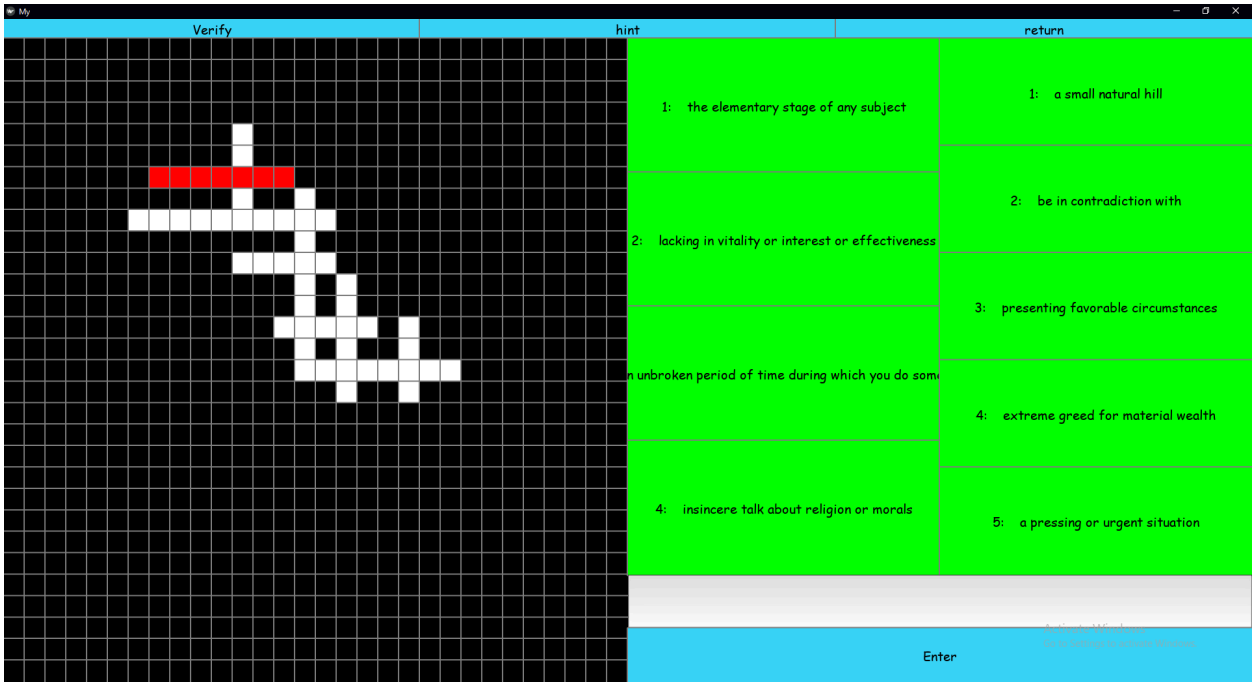
```
def highlight(self, *largs):
    print(self.manager.get_screen("settings").palette)
    print(largs)
    print(self.selected[0])
    if self.selected[2] == "down":
        for m in range(self.selected[1]):
            self.my_buttons[self.selected[0] + 30 * m].background_color = self.manager.get_screen("settings").palette[3]
    if self.selected[2] == "across":
        for m in range(self.selected[1]):
            self.my_buttons[self.selected[0] + m].background_color = self.manager.get_screen("settings").palette[3]

    if largs[0][2] == "down":
        for l in range(largs[0][1]):
            self.my_buttons[largs[0][0] + 30 * l].background_color = self.manager.get_screen("settings").palette[2]

    if largs[0][2] == "across":
        for l in range(largs[0][1]):
            self.my_buttons[largs[0][0] + l].background_color = self.manager.get_screen("settings").palette[2]

    self.text_input.text = ""
    self.selected = [largs[0][0], largs[0][1], largs[0][2]]
```

Test ID 25

Tested with the improved function now:







As you can see the function is now working as intended and only one clue is highlighted at a time. Success

Tutorial

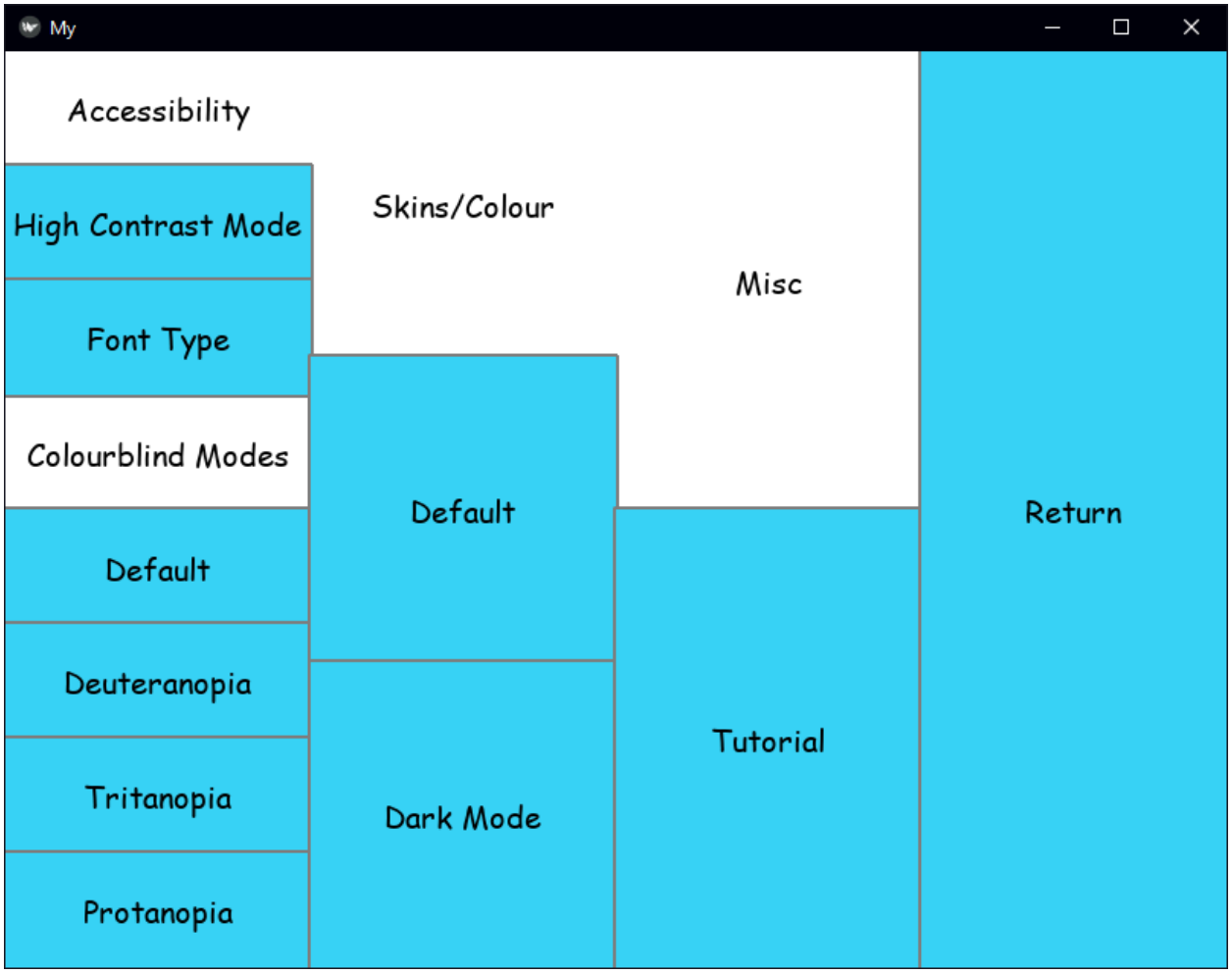
To make my program as accessible as possible it needs a tutorial for new users to read.
I decided to put the button inside the settings menu as that would be the first place any lost users would look to find a tutorial.

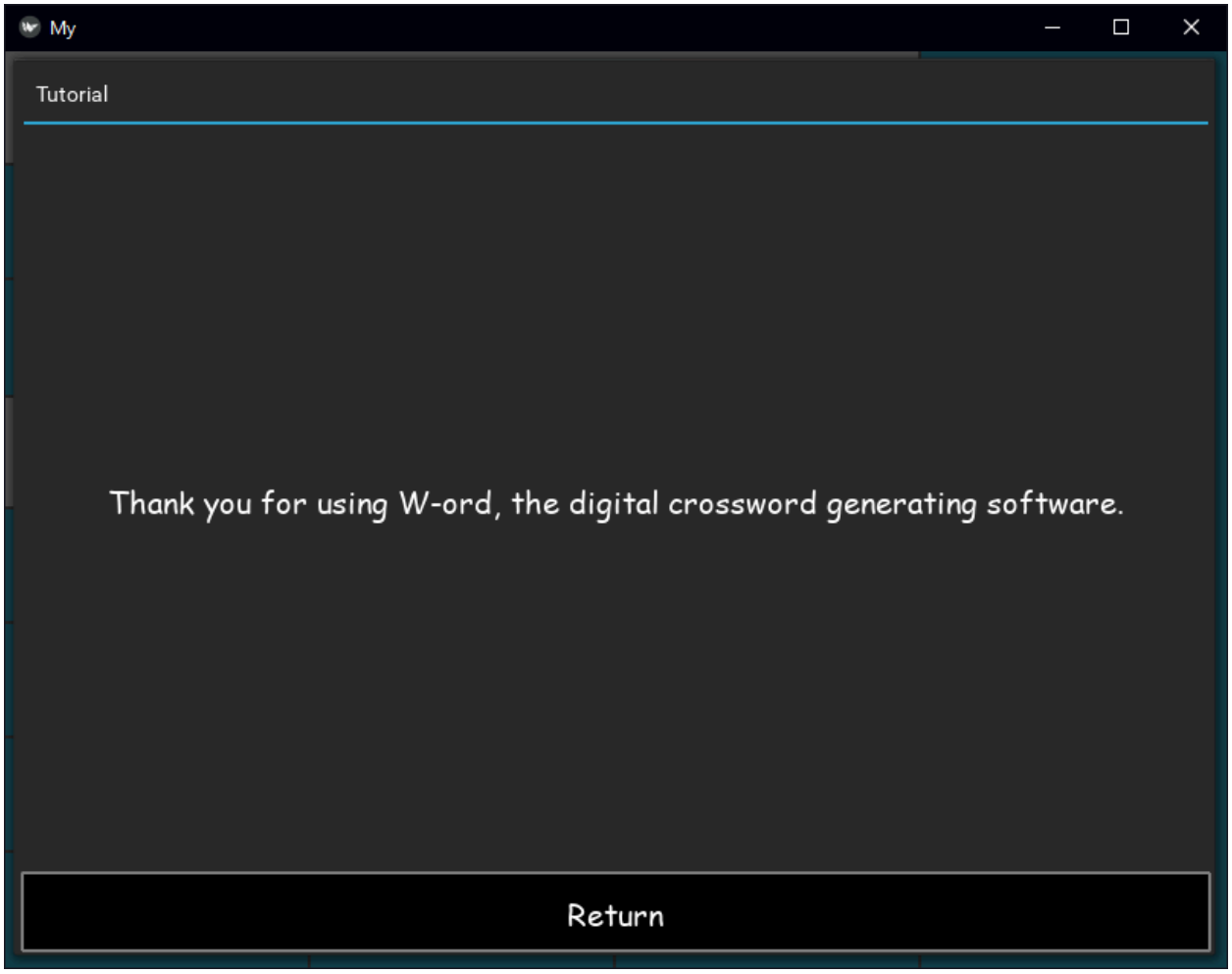
For the purpose of temporarily displaying text across the whole screen I went to use popups which is a module in Kivy. It functions similarly to a screen, the main difference being it needing a dismiss button. The following code is my approach:

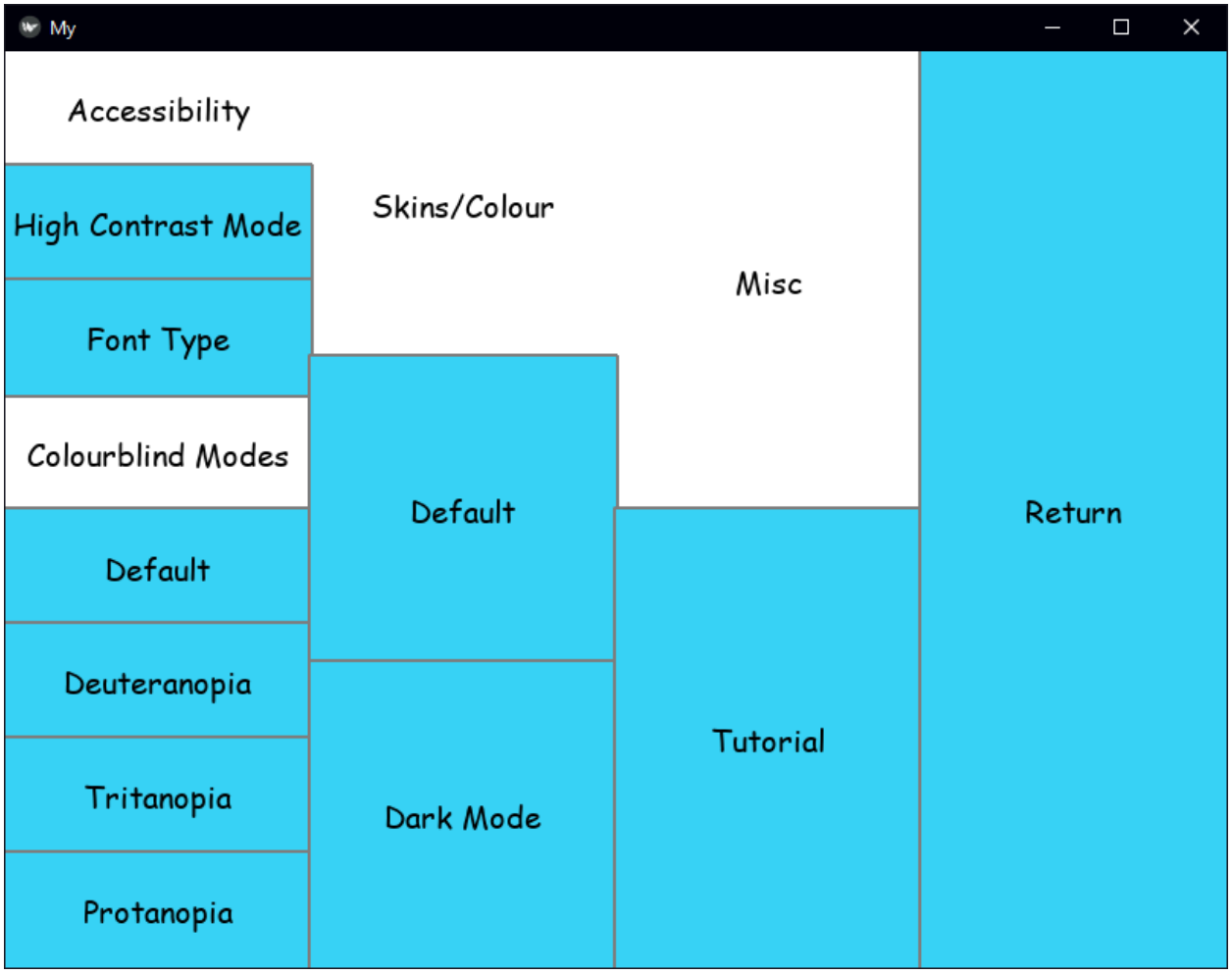
```
def tutorial(self):     # tutorial popup
    box = BoxLayout(orientation="vertical")
    box.add_widget(Label(text="Thank you for using W-ord, the digital crossword generating software.",color=(1,1,1,1)))
    btn1 = Button(text="Return",size_hint=(1, 0.1),background_color=(0,0,0,1),color=(1,1,1,1))
    box.add_widget(btn1)
    popup = Popup(title="Tutorial",content=box)
    btn1.bind(on_press = popup.dismiss)
    popup.open()
```

Test ID 27

Upon press the popup should appear and display correctly. Inside the popup pressing the “Return” button will close the popup and leave the user where they left off, the settings page.


My
Accessibility
High Contrast Mode
Font Type
Colourblind Modes
Default
Deuteranopia
Tritanopia
Protanopia
Skins/Colour
Default
Dark Mode
Misc
Tutorial
Return


My
Tutorial
Thank you for using W-ord, the digital crossword generating software.
Return



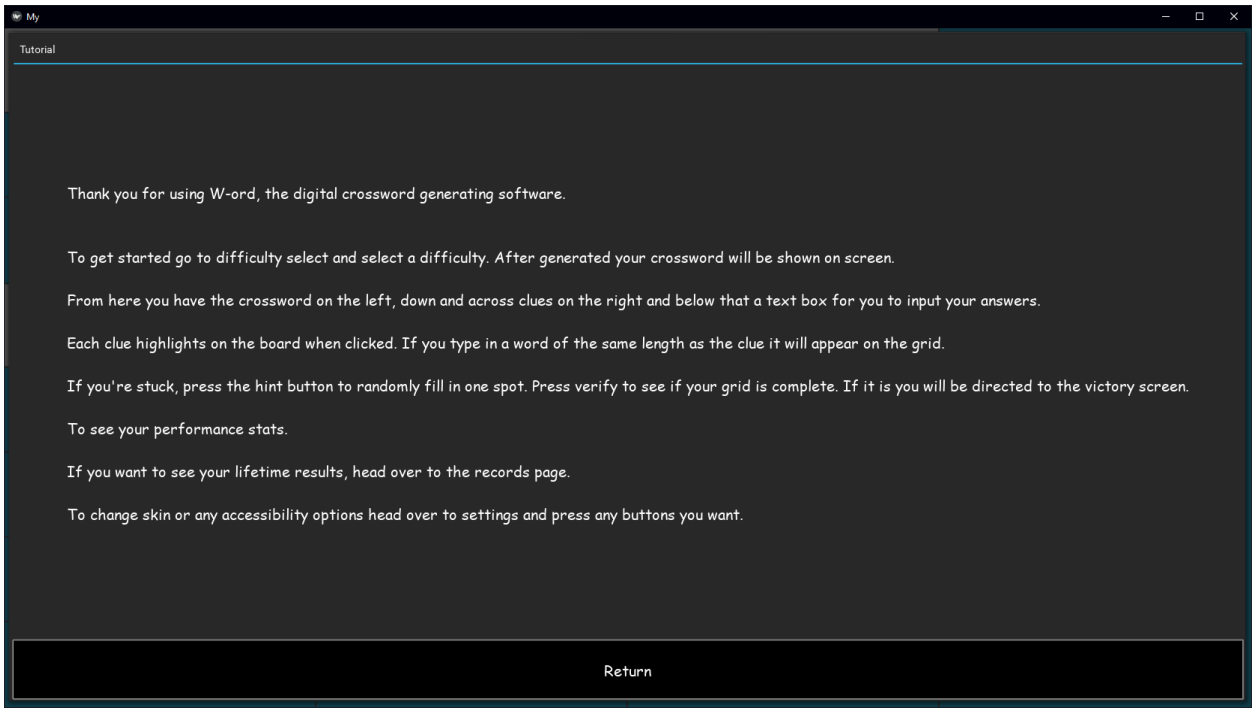
All results as expected: success.

Now that functionality works I need to design the tutorial itself. The main focus should be on teaching users how to input to the crossword given that is the most complex action. It should also teach them what the settings and records pages are for.

The tutorial is as follows:
"Thank you for using W-ord, the digital crossword generating software.\n\n\n"
“To get started go to difficulty select and select a difficulty. After generated your crossword will be shown on screen.\n\n"
"From here you have the crossword on the left, down and across clues on the right and below that a text box for you to input your answers.\n\n"
"Each clue highlights on the board when clicked. If you type in a word of the same length as the clue it will appear on the grid.\n\n"
"If you're stuck, press the hint button to randomly fill in one spot. Press verify to see if your grid is complete. If it is you will be directed to the victory screen.\n\n"

"To see your performance stats.\n\n"
"If you want to see your lifetime results, head over to the records page.\n\n"
"To change skin or any accessibility options head over to settings and press any buttons you want."

When put into the popup function it displays as follows



Overall I am happy with this function

Summary of Iteration 2:

This iteration has accomplished its objectives being to add functionality to all the buttons. It has also satisfied all of the test plans as established in the design phase. By doing such I can consider my program complete.

Evaluation

Post Development Testing

To prove that my solution is robust and functioning as intended I gave my initial stakeholders and 3 other potential users my application to test and play around with. I provided them each with the following table of potential things they can do and asked them to give feedback whether the features were working and their general thoughts on each feature. I asked my 2 original stakeholders in particular to try and break my program and record evidence.

Question	**Answer**	**Comment**
1. Does the application open?		
2. Do the navigational buttons work ie records goes to records page, crossword select goes to crossword select		
3. Does pressing a difficulty button generate a functioning crossword?		
4. Do you notice a difference between each difficulty		
5. Do the clues respond when clicking on them and highlight the grid		
6. Are the clues accurate to the words they map to? Do you think they're fair?		
7. Does entering text into the text box work?		
8. Is the crossword page easy to understand and use?		
9. Does the hint button work		

10. Does the verify button work		
11. Is the data displayed on the victory screen accurate to the data for the crossword you just completed?		
12. Does the records page correctly display your information?		
13. Does the records page display relevant information?		
14. Does the reset data button work? You need to refresh the records page to confirm this		
15. Do the buttons to change skin work?		
16. Are you satisfied with the selection of skins available?		
17. Does colourblind mode work? If you are colourblind is it suitable?		
18. Does high contrast mode work and is it suitable?		
19. Does Font Change change the current font?		
20. Does the tutorial button work? Is it informative enough?		
21. Is the performance of the program satisfactory?		
22. Closing feedback/unmentioned bugs?		

User responses:

Question	Answer	Comments
1. Does the application open?	Yes [6]	
2. Do the navigational buttons work ie records goes to records page, crossword select goes to crossword select	Yes [6]	-I like the simplistic design -I like how similar it is to the original draft (design UI)
3. Does pressing a difficulty button generate a functioning crossword?	Yes [6]	
4. Do you notice a difference between each difficulty	Yes [5] No [1]	-I found towards the harder end they were quite similar -I noticed a difference however I think it'd be better to have words from easier difficulties in harder ones to ease the learning curve
5. Do the clues respond when clicking on them and highlight the grid	Yes [6]	-That's a neat feature there it really adds clarity -Wow this make the program feel alive
6. Are the clues accurate to the words they map to? Do you think they're fair?	Yes [6]	-Occasionally some clues would overlap outside of the text box. This might just be because I have a very small screen
7. Does entering text into the text box work?	Yes [6]	-I wasn't able to break the program here or pass any illegal inputs. It's very well validated and always outputs lowercase words properly -I like how I can press enter or use the button to enter words
8. Is the crossword page easy to understand and use?	Yes [6]	
9. Does the hint button work	Yes [6]	-This functions but maybe the hint button could change one of the actual hints for the

		words. This would both be more helpful and interesting
10. Does the verify button work	Yes [6]	
11. Is the data displayed on the victory screen accurate to the data for the crossword you just completed?	Yes [6]	-The data displayed is suitable however I believe it could benefit from showing the user their average time so they can compare it to what they just achieved
12. Does the records page correctly display your information?	Yes [6]	
13. Does the records page display relevant information?	Yes [6]	-I really like how it records the fastest time for each difficulty. Really allows the user to see their improvement
14. Does the reset data button work? You need to refresh the records page to confirm this	Yes [6]	
15. Do the buttons to change skin work?	Yes [6]	
16. Are you satisfied with the selection of skins available?	Yes [5] No [1]	-Although the skins created are very good I'd still like a few more
17. Does colourblind mode work? If you are colourblind is it suitable?	Yes [6]	-I am colourblind and I appreciate the effort put into this. I have no problem seeing anything in the program.
18. Does high contrast mode work and is it suitable?	Yes [6]	
19. Does Font Change change the current font?	Yes [6]	
20. Does the tutorial button work? Is it informative enough?	Yes [4] No[2]	-too long -too many words
21. Is the performance of the	Yes[6]	-I was able to run this on my

program satisfactory?		old laptop with no problems. It would slow when generating the crossword. I only noticed this because there was no loading screen -Ran smoothly every time on my i5 processor, 4Gb RAM laptop
22. Closing feedback/unmentioned bugs?		

These will be referenced as beta test 01 for example

Success Criteria

1 : Home Page

This objective has been fully met. Test ID 28 proves all navigational buttons to function hence meeting the criteria. All users were able to navigate it (beta test 02). Some stated they liked the simplistic design and that it stayed consistent with the original UI design I showed them.

2: Difficulty page

This objective has been fully met with 4 different difficulties. See tests ID 10. These prove the difficulty page as successfully working under all circumstances. The beta testers further backed this up in example 03 where all 6 testers had no errors with generation.

Similar to the Home Page my stakeholders were also positive about the layout and that it was exactly what I'd shown them a few weeks prior.

One user from beta test 04 did not notice a difference between difficulties. This may be due to them having a different vocabulary to the other testers but the general consensus is that the change is noticed.

One raised the suggestion that harder difficulties should include a few but not many words from lower difficulties to decrease the steep difficulty increase. I see where they're coming from as it

would make the harder ones more familiar to the user. Any future updates to this program may wish to implement this. The only downside would make it harder to see if test data is working.

3: Stattracking system

This objective has been fully met. Tests ID 12-16 Show that all the initial listed fields are recorded and Test IDs show that they are all properly validated. Stakeholders were satisfied with the choice of data I recorded stating: “given the nature of your program, you’ve recorded all relevant and data of interest. They also noted this system in particular to be very robust and maintainable.

4: Records page

This objective has been completely met. Tests ID 17-20 support this. Errors found in these were taken and fixed so that the tests could be rerun and were successful. The tests also show that the data displayed is validated and formatted to the correct standards. Beta test 12 shows that all testers agree that the data is displayed correctly which indicates the system is validated properly. Beta test 13 shows that all testers agreed that the data stored is relevant and in particular the fastest time for each difficulty adds more depth to the application.
Based on this any future improvements would likely come as a layout change but that is subjective to whomever is using it.

5: Functioning UI

This objective has been fully met. Tests ID … support this and the validation done to be robust and reliable. Stakeholders stated that this page did exactly what it set out to do and are content with it. Any future improvements would likely come as a layout change but that is subjective to whomever is using it.

6: Crossword page
This objective has been fully met. Beta test 8 shows that all testers easily understood and used the crossword page.

7: Down and across clues
This objective has been successfully met. Test ID 25 proves that the buttons represent the correct cells on the grid. Beta tests 5 and 6 indicate that the clues are both responsive and accurate to their mapped words.

One user had issues where the clue text would overextend outside of its box. Although stated they had a small screen future iterations should do more to prevent errors like this.

8: Input Box

This objective has been fully met. Tests IDs 01-05 show user input being applied to the grid if a suitable clue is selected. Further testing is done during iterative development to ensure the data added to the grid is validated and accepts either lower or upper case and converts to lower. Beta test 7 supports all of this as nobody was able to break the text box and it was stated that they always output lowercase words. From this test stakeholders also appreciated the quality of life feature of pressing enter to enter words or using the button.

Future iterations would have little to improve on.

9 : Highlight grid
This objective has been successfully met and proven by Test ID 25. Test IDs 21 and 22 also prove it to function with skins and colour modes. Based on beta test 05 stakeholders liked this feature, stating that it adds much more life to the program and especially with younger players will appeal to them. A future iteration would likely do the same.

10:Hint button

This objective has been successfully met. Test ID 09 proves that the hint button functions as intended and is accurate to the real answer. Beta test 09 shows stakeholders were happy with this. One suggested a future iteration might provide hints as an alternate definition to the current clue. I agree that this is a great suggestion and a future iteration would benefit from having it, maybe alongside the current one used.

11: Verify button
This objective has been successfully met. The verify button works properly and returns success only when the entire crossword is correct, as proven by Test IDs 06-08 It is thoroughly validated and proven here too. Based on beta test 10, stakeholders had no hiccups with this button and found it worked properly for all of their crossword attempts whether that be with a correct or incorrect input. A future iteration might want to highlight which grid cells are incorrect. I initially thought this would make it too easy but after experimenting I agree this would be a nice feature and make the crossword more accessible to less skilled players.

12: Victory screen
Successfully met. Test ID 26 proves it is functional and further testing was done to ensure the data displayed is up to date and accurate at page. Beta test 11 shows that stakeholders were

satisfied with its appearance and feedback upon completion. One tester suggested a future improvement might give more precise feedback such as displaying a comparison to the average time completed so that users can see how their latest crossword was normal. I fully agree with this as it adds much more depth and accomplishment to the program and any future iteration should attempt to implement it.

13: Settings page
partially met. Proven to be functional and provide all features necessary by tests ID 21-24 and 28. Based on beta test 16, 5 out of 6 stakeholders were satisfied by the customisable options available. The one who wasn't stated that the options in place were very good but they just wanted more of them.

Beta test 17 shows that all users were able to select colourblind mode with no issues. Out of the group I specifically picked someone who I know to be colourblind for this test. They're the most important factor for this test and having their feedback being that they have no problem seeing anything in the program makes the feature a success. I could only test for one type of colourblindness () but the method of using a monochrome palette is proven to benefit most types.

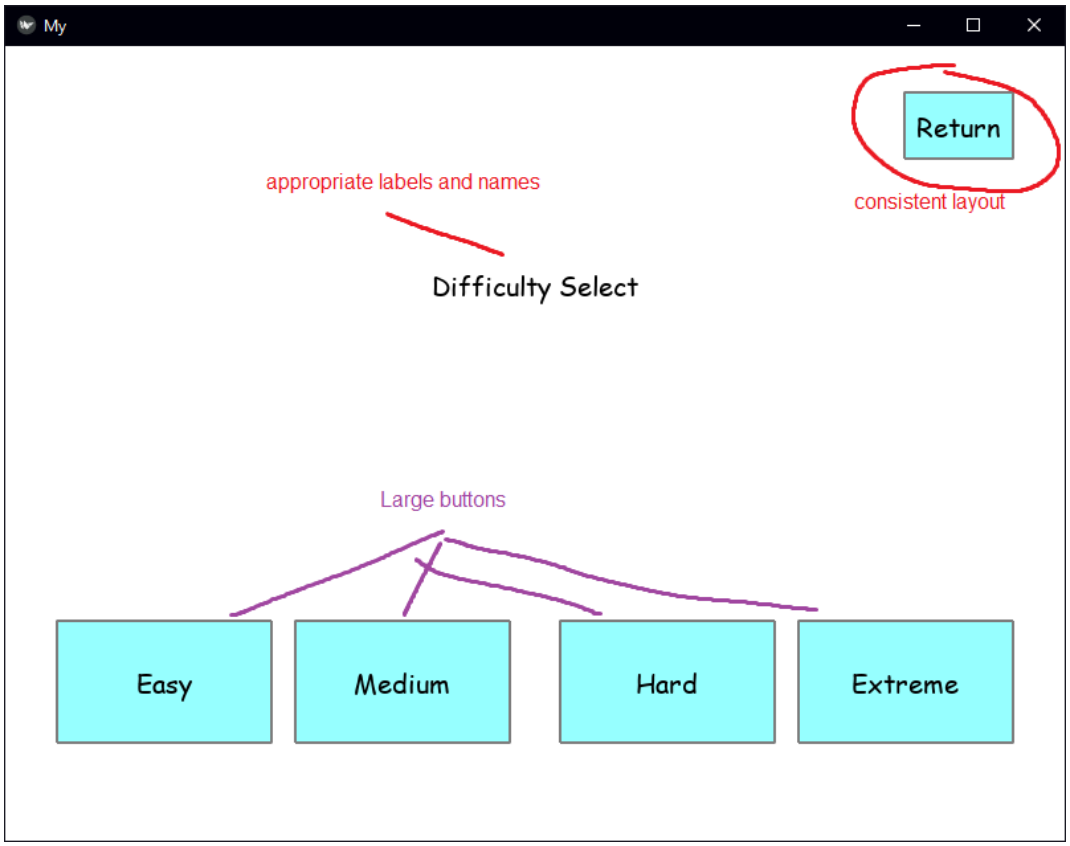
Alongside accessibility font changes and sizes. High contrast mode was designed similarly to Windows 10's version, ensuring each widget is as clear and distinct as possible. Beta test 18 showed it to be working for all users and rated suitable.

The only thing missing from here is toggling the text size. This is because it would interfere with other widgets too much, especially on smaller screens which I tested on. Text size would also slightly differ depending on font selected which further skewed my attempts. Stakeholders I consulted weren't too inconvenienced, stating that the default size is about as large as I could make it without running into any errors.

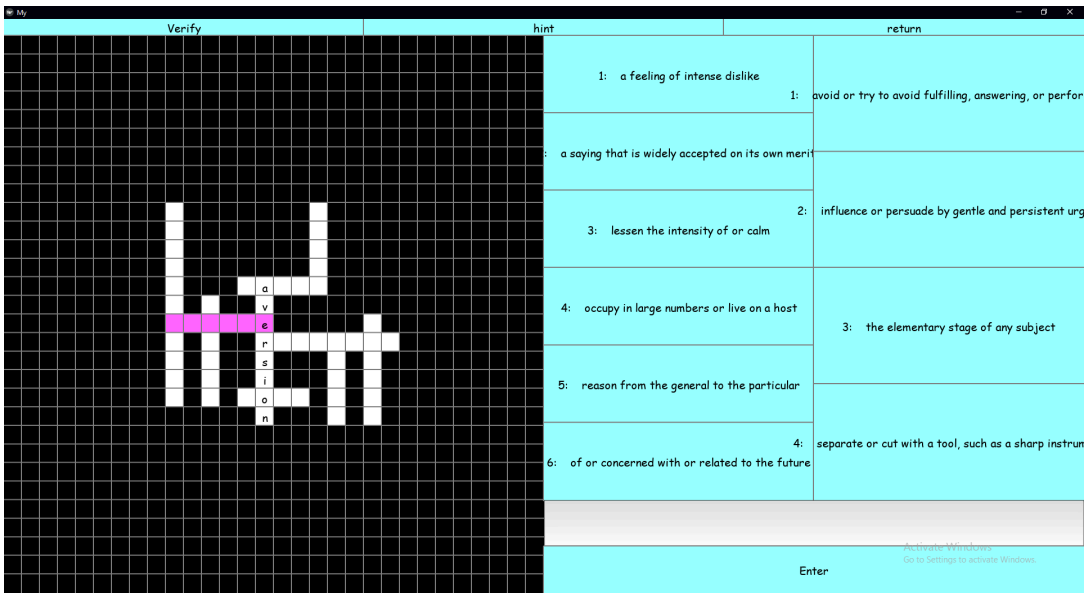
14: Informative Tutorial
Fully met. Test ID 28 provides evidence of the tutorial being opened and closed successfully from the settings menu. Feedback from my stakeholders in beta test 20 was generally positive however some said it may have been too lengthy. Some disagreed on this which led me to believe it to be personal preference. At the end of the day my tutorial provides clear instructions on how to operate the software and it's better to have too much information than too little. A future iteration might wish to make the tutorial more concise and digestible.

Usability Features

My buttons and labels are large and labelled appropriately, making navigation between screens very easy.

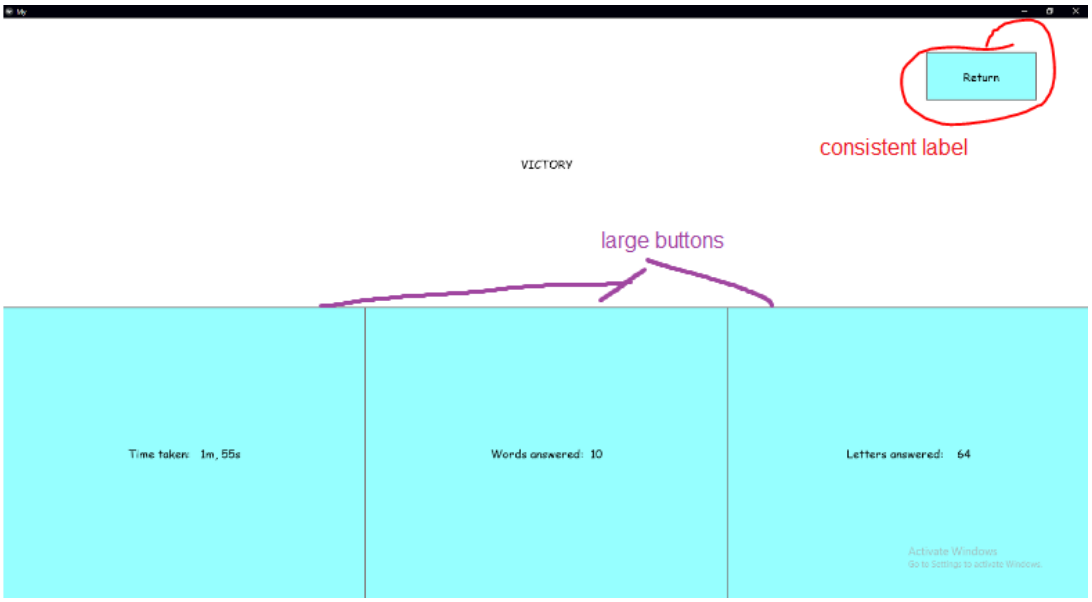


Each page has a consistent layout consisting of the Return button in the top right and the content underneath usually split into vertical columns.

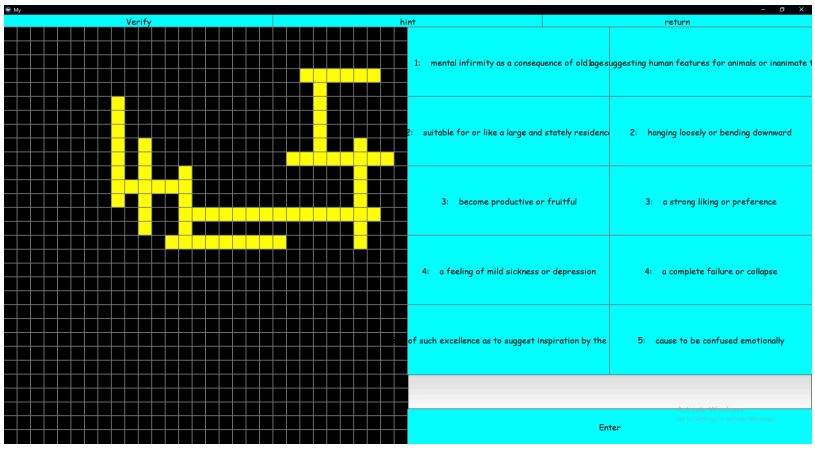


The number of buttons on each screen is kept to a minimum to stay in line with that simplistic design.

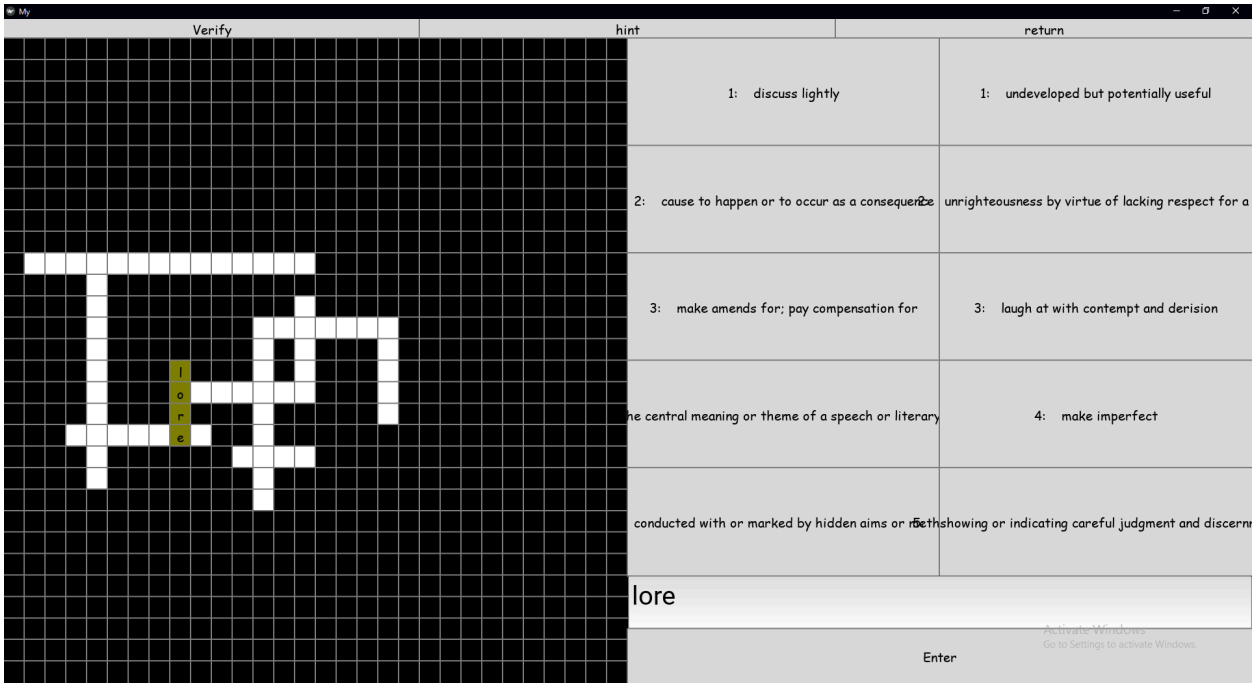
Figure on left has vertical column layout



High contrast mode can be selected for those with low vision. This enables more people to enjoy my product.



Colourblind mode can also be selected to branch out to an even larger audience



Finally there is a font toggle button in settings to make it even easier for people to access my program. The font is set to Comic Sans as default and if people wish to change it they can go to settings and toggle Arial

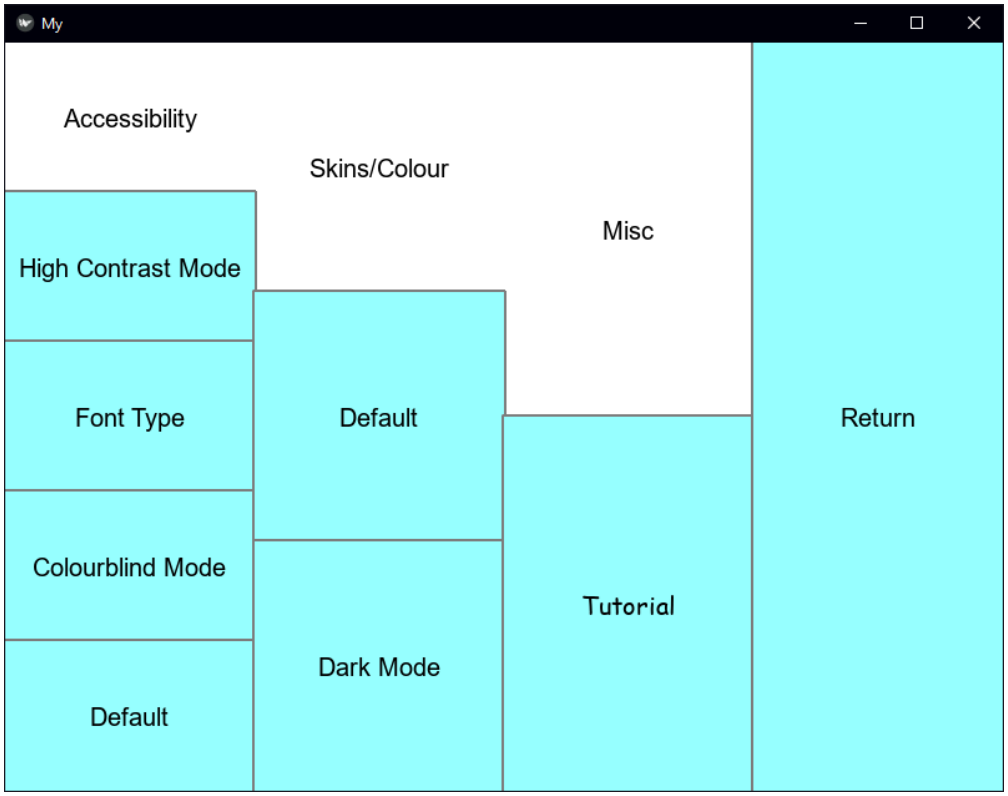


Figure above has appropriate button names and large buttons

Maintenance

By utilising this document my application is made maintainable. Individual algorithms are split into different functions which are appropriately named. The bulk of the code is split into classes based on which screen of the UI they are relevant to which helps to keep data and variables organised and together in one place. Inheritance is used to minimise programming power needed and make my solution more efficient.

My solution also has comments where necessary, labelling any key features or processes which anyone attempting to update the code might need.

Non-GUI related functions and procedures are compiled into libraries to keep the main program clean and easier to read. Another benefit of this is that they are already stored in separate files therefore any improvements or fixes needed can be done within said file with artificial test data.

Limitations

Based on user feedback and the evaluation of the success criteria I have identified a multitude of different areas which I could improve upon as listed above.

My project has fulfilled the majority of the success criteria and undergone beta testing by multiple people to ensure it’s robust. This coupled alongside the evaluation lead to the conclusion that this project was successful.