AS3.46 (91637) v Develop a complex computer program for a specified task

Resource title: Student Worksheet and Answer Booklet

1. Type your name in the header of this document.
2. You have until the **end of Week two** in **Term 4** to complete the assessment. This time includes school periods, out of class time and the term 3 holiday period.
3. The test is conducted under **test conditions** in class (no talking).
4. Name your **Student Worksheet** and your **Python code sheet** "first initial, surname + AS3.46 + version number". EG **GHall_AS3.46_v3**
5. Save your completed **Student Worksheet** and your **python code** to a folder in your H: drive named "first initial, surname + AS3.46". EG **GHall_AS3.46**
6. Save your documents (including code) at regular intervals to the H: drive.
7. To hand in the assignment, you need to copy your folder to the appropriate folder in the **13CSI Dropbox**.

Learner		Name		Signature
	Luke Hawinkels		L.A.H	
Result	**Not Achieved**	**Achieved**	**Merit**	**Excellence**	Date
Assessor		**Gordon Hall** Name		Signature

Resubmit	**Not Achieved**	**Achieved**	**Merit**	**Excellence**	Date
Assessor		**Gordon Hall** Name		Signature

Your Project

Evidence

Collect and comment evidence (screenshots) documenting how you have implemented the design and build of this project. The evidence should show that you have:

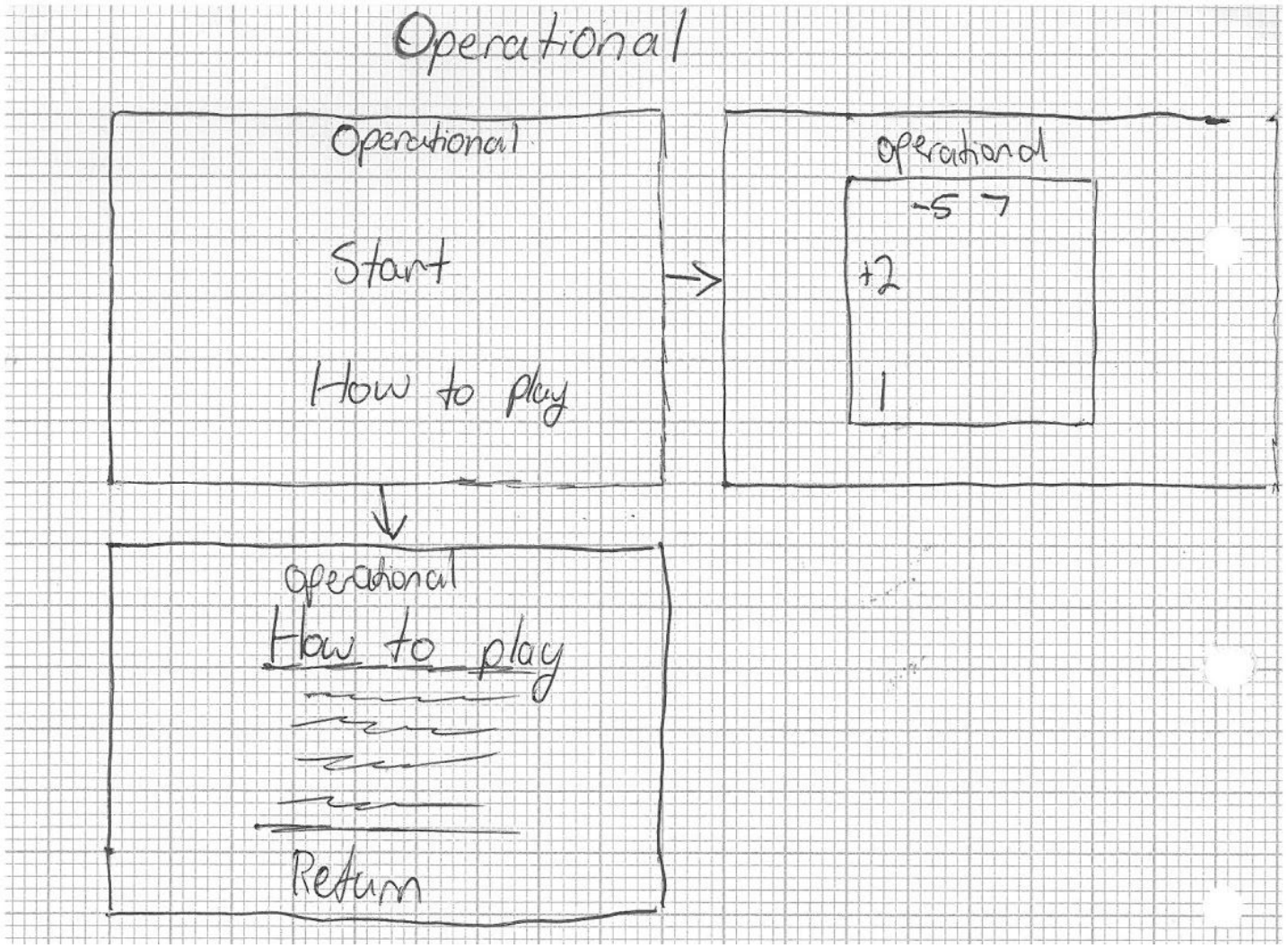
1. Developed and followed a **plan** (**GUI** (Graphical user interfaces) and **Flow chart**) for building your program by saving successive versions of the **GUI, Flowchart *(Modular Structure)*** and **Python code** with added complexity in each version.
2. Worked in an organised and efficient way, using time effectively and not using a trial-and-error approach unnecessarily.

Part 1 - Planning the program

Commented Screenshots showing your Graphical user interface version(s).

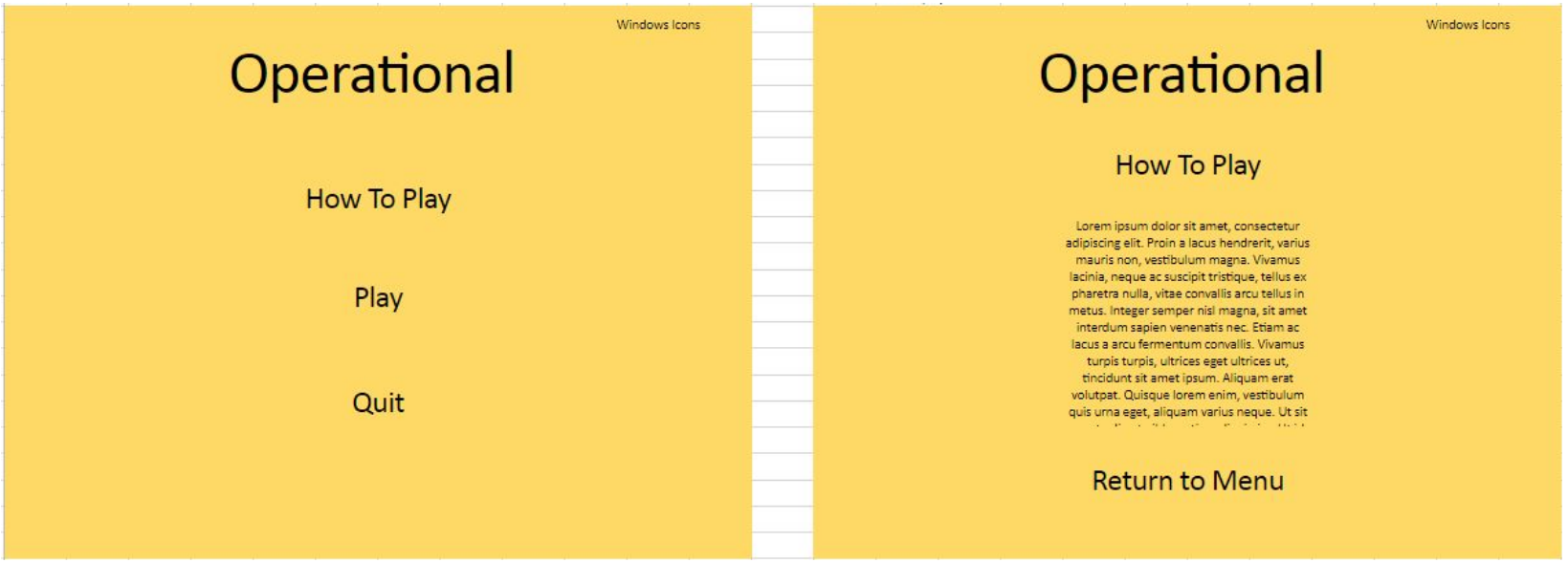
Graphical user interfaces - Version(s)

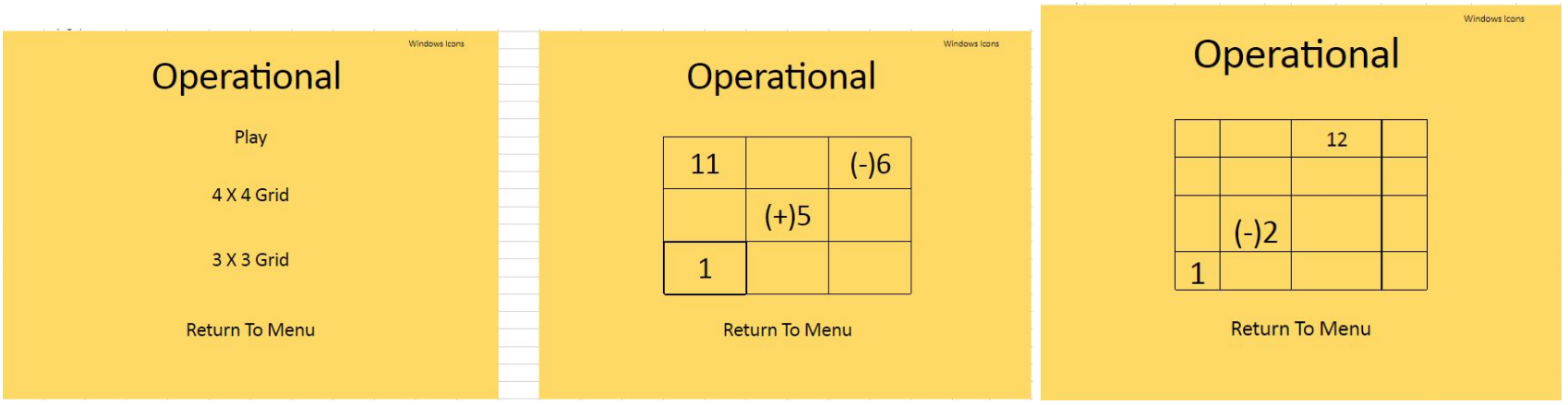
Version 1



https://drive.google.com/open?id=0B8a2I31inGKWVG9OdXMxNnFJNkwxaEloNDVCano1dExDdW9J

Version 2





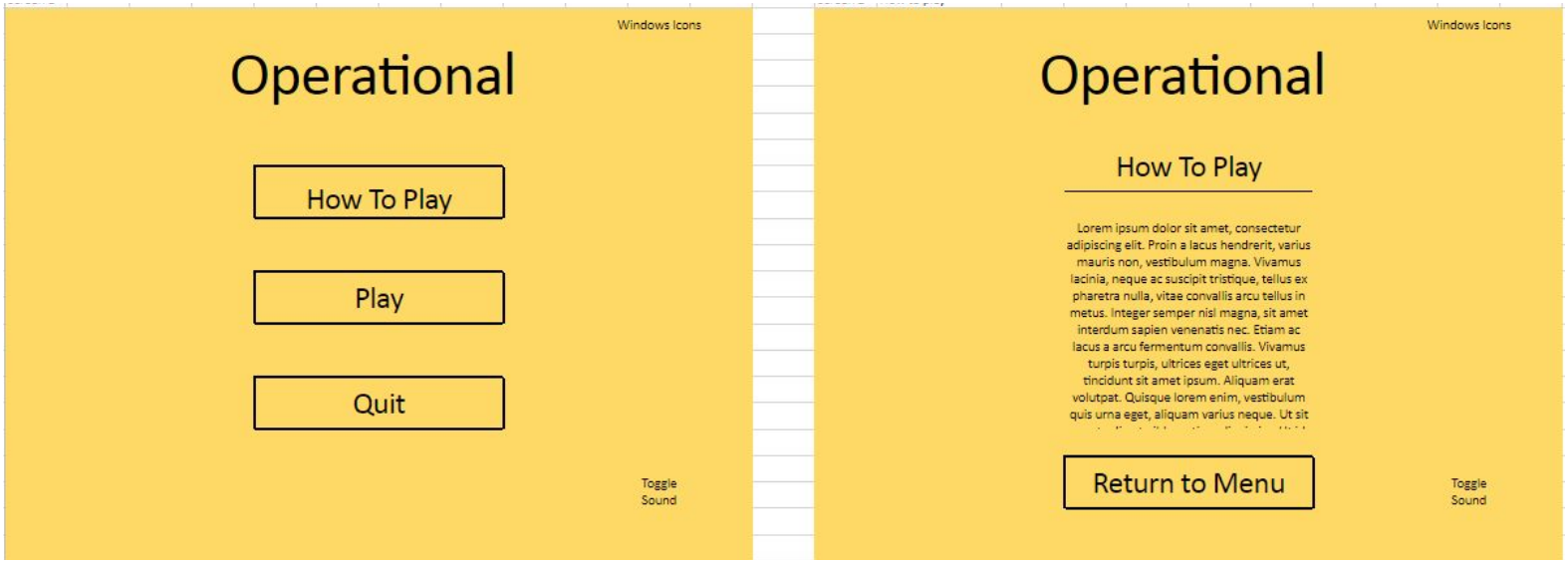
This version includes the following changes…

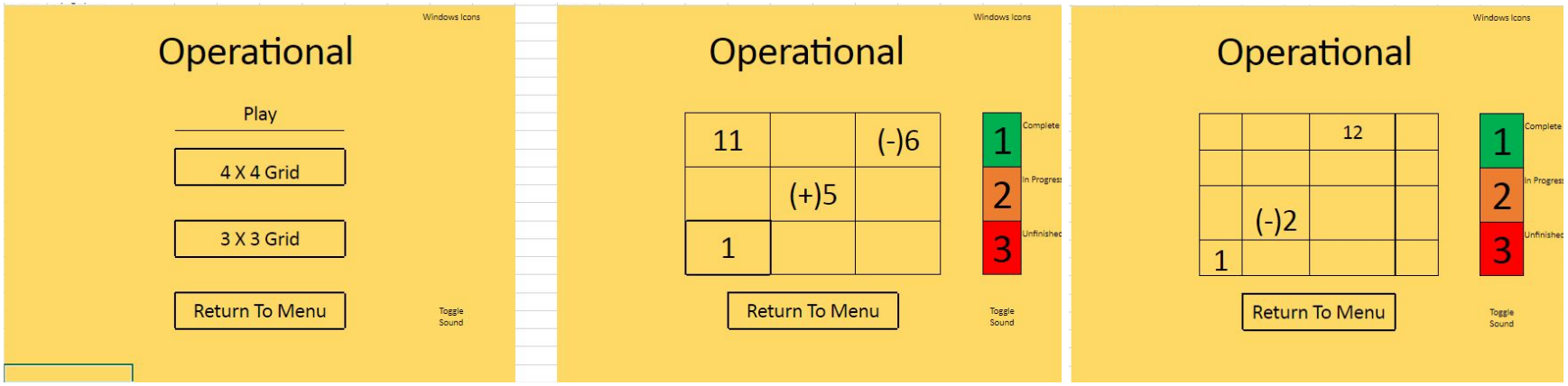
- A placeholder for the standard windows icons has been added.

- Options for different Grids have been added

NP: Squares and windows icons (top right) are inaccurate due to the limitation of excel and will be correct in the final program

Version 3



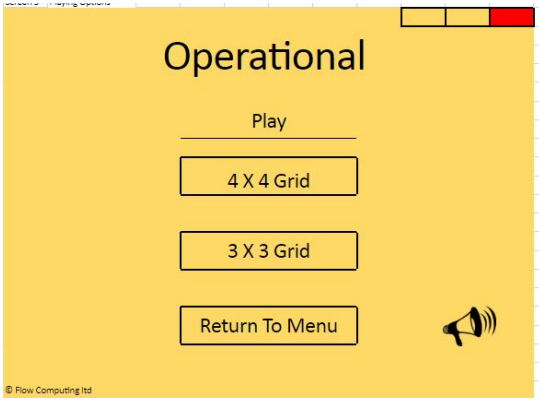


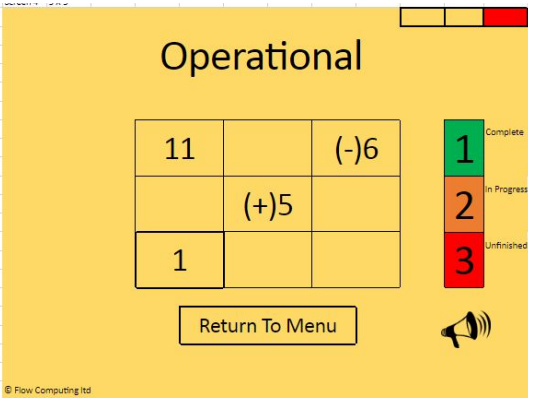
NP: Squares and windows icons (top right) are inaccurate due to the limitation of excel and will be correct in the final program

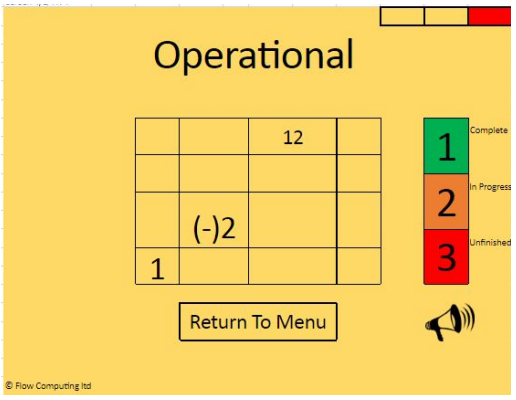
This version includes the following changes...

- Buttons now have borders
- There is a progress bar on the side
- There's an option to toggle sound
- How to play has a line under it

Version 4







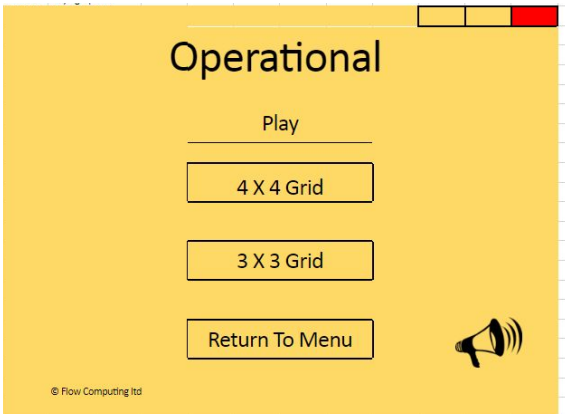
To view all screens please click this link. https://1drv.ms/x/s!AoL5Qpy3NBwXhvsw6F1Yl7v5rc_hiQ

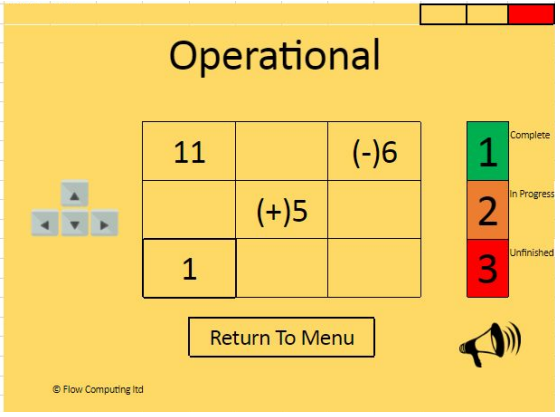
NP: Squares and windows icons (top right) are inaccurate due to the limitation of excel and will be correct in the final program

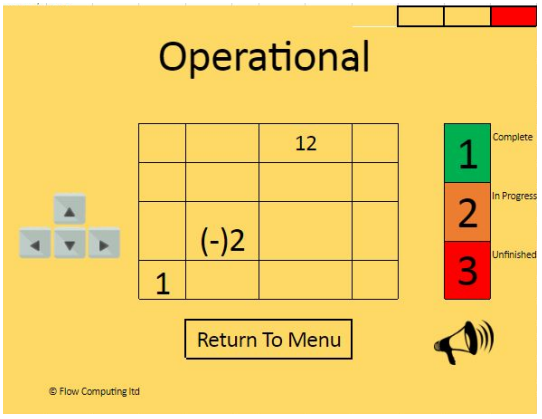
This version includes the following changes...

- Megaphone for sound toggle
- More accurate placeholder windows icons (It is inefficient and time consuming for me to create accurate one in excel).
- Line under the play heading
- Company information added

Final Graphical user interfaces







NP: Squares and windows icons (top right) are inaccurate due to the limitation of excel and will be correct in the final program

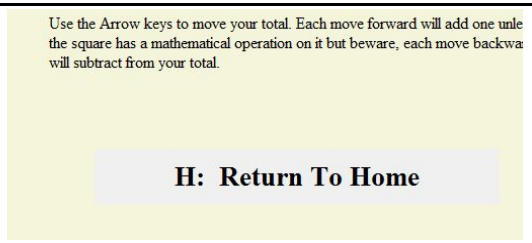
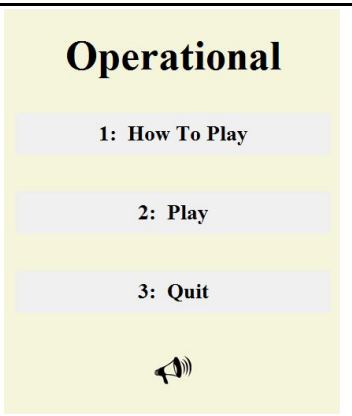
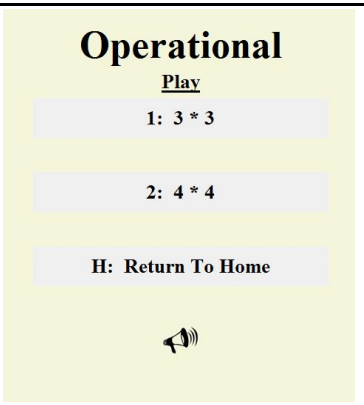
This version includes the following changes...

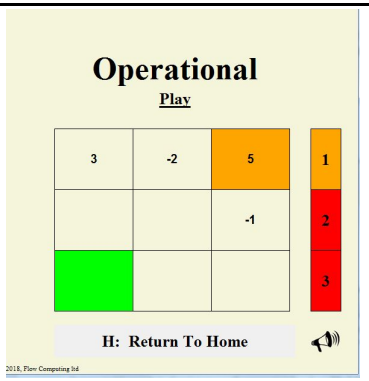
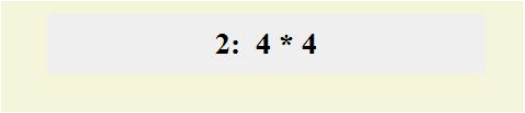
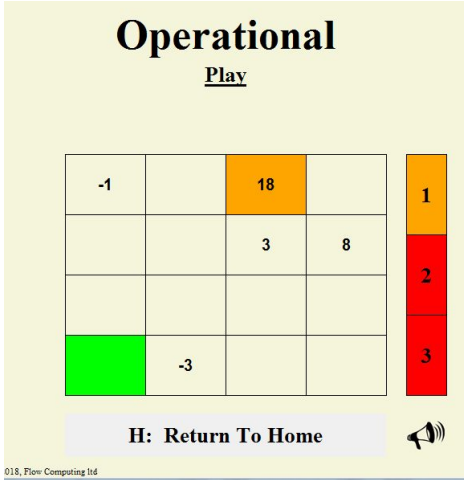
- flow computing logo has been moved to add more whitespace
- Arrow keys have been added to the play screens. These are both instructional and Functional.

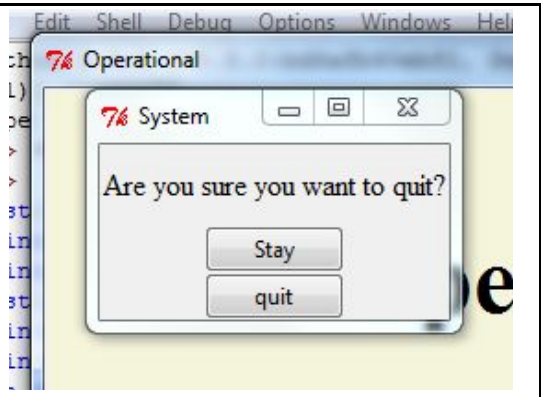
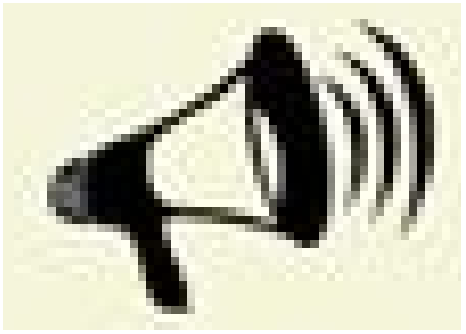
NP:The three by three version works perfectly, the four by four version is experimental and and be buggy, a five by five could be offered in future.

NP: Due to difficulty and time constraints I was unable to add the keyboard arrow key functionality to my program. This could potentially be added in later revisions.

Event generating and handling components	Event generating	Event generated
The user clicks on the entry field which allows the user to enter their name into the field. This is done to allow a degree of personalisation to the program.		
The user clicks on the Ok button which checks the name and then continues the program. This is done to allow the user to continue through the program and prevent frustrations.		Luke **Operational**
The user **clicks on the instructions button** which takes them to an instructions screen. This is needed because the user needs to know how to play the game. The same event can also be triggered when the **user pushes 1 on their keyboard** on the appropriate screen.	1: How To Play	**Operational** How To play To win land on the target square with the correct total at the right time.... Use the Arrow keys to move your total. Each move forward will add one unless the square has a mathematical operation on it but beware, each move backwards will subtract from your total.

The user **clicks on the Return to home button** and they are taken back to the homescreen This is done because the user needs a way to backtrack if they clicked on something they didn't want to. the same event can be triggered when the user **pushes h on their keyboard**.		
The user **clicks on the play button** so that they can proceed with the game. This button is needed so that the user can progress to the next menu. The same event can be triggered when the **user pushes 2 on their keyboard** on the appropriate screen.		

The user **clicks on the 3 * 3 button** which loads up a three by three grid. This is needed so that the user is able to start the game. The same event can be triggered when the **user pushes 1 on their keyboard** on the appropriate screen.		
The user **clicks on the 4 * 4 button** which loads up a four by four grid. This is needed so that the user is able to start the game. The same event can be triggered when the **user pushes 2 on their keyboard** on the appropriate screen.		

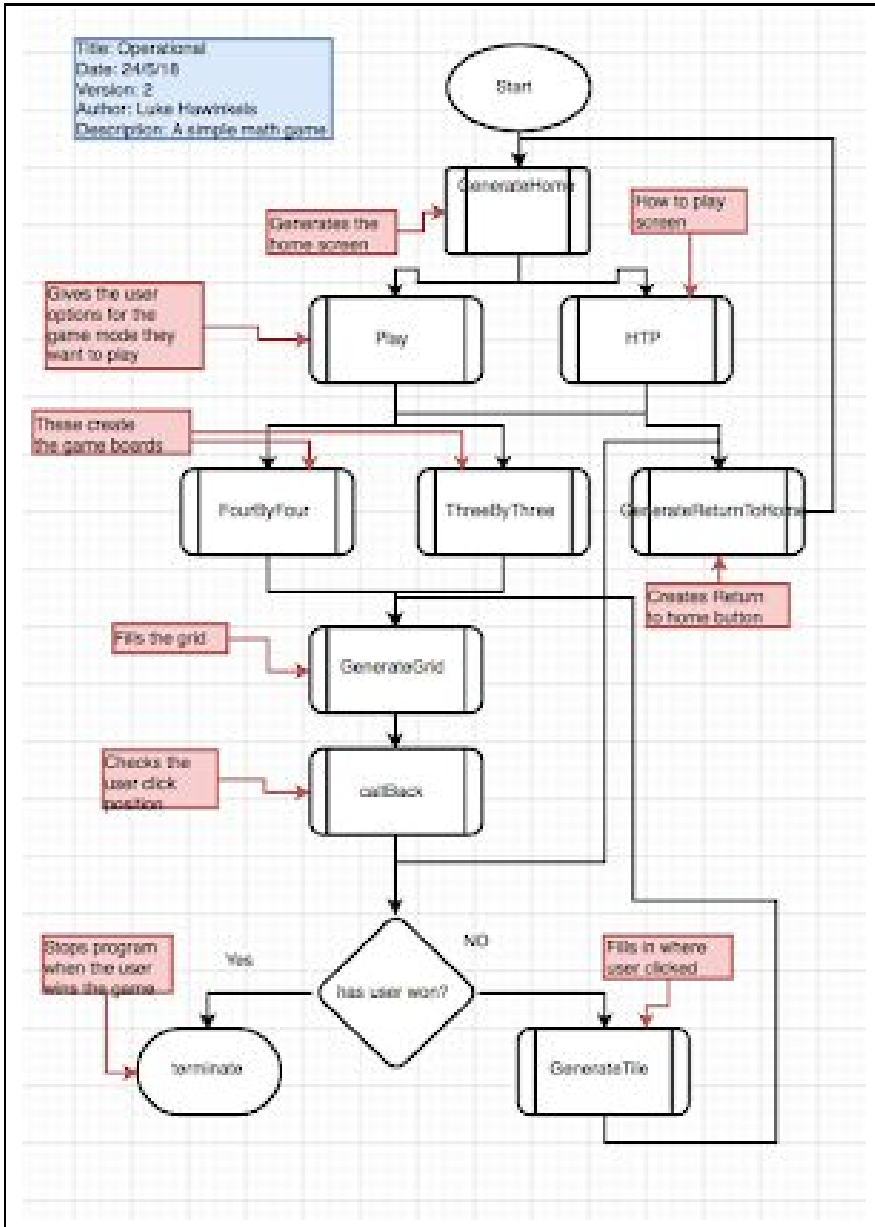
The **quit button** exits the game. This is needed so that the user can quickly and easily exit the game if they need to. This event can also be triggered when the **user presses 3 on their keyboard** at any time.	3: Quit	
The music button plays and pauses the music. It is necessary because music is an optional extra and the user should not be forced ro have it play.		

You may have more or less than 3 versions (these may be attached at the end of the report rather than screen shot here. If attached, still comment on any changes and additions).

*Commented Screenshots showing your well commented flowchart (**Modular structure**) version(s).*

Commented Screenshots showing your flowchart version(s):

Version 1


Title: Operational
Date: 24/5/18
Version: 2
Author: Luke Hawinkels
Description: A simple math game
Start
GenerateHome
Generates the home screen
How to play screen
Gives the user options for the game mode they want to play
Play
HTP
These create the game boards
FourByFour
ThreeByThree
GenerateReturnToHome
Creates Return to home button
Fills the grid
GenerateGrid
Checks the user click position
callBack
Stops program when the user wins the game
Yes
has user won?
NO
Fills in where user clicked
terminate
GenerateTile

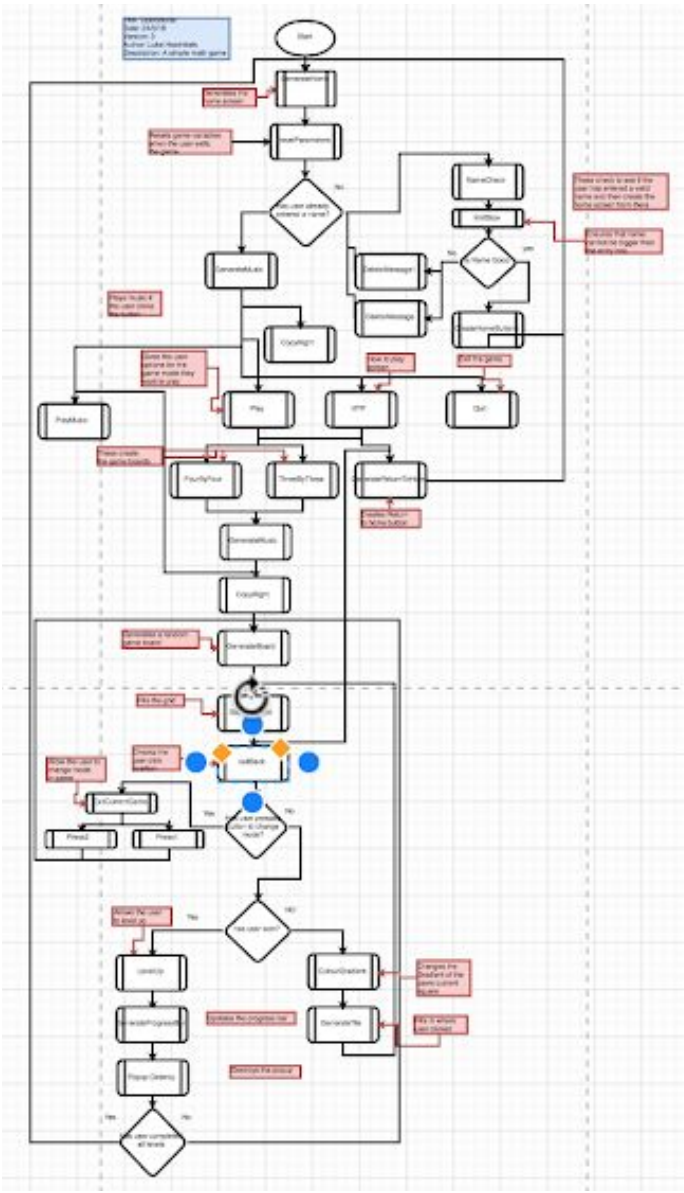
View here: https://drive.google.com/file/d/125QDrPIjAEjPmWe2wKtkaxUUUm5j8g5T/view?usp=sharing

This version of the program is the most basic. It provides the following….

- A home screen
- Instructions
- Quit functionality
- Two different game board types
- One game board of each type
- A basic system for showing where the user is

This is the first version of the program. It's only purpose is to be functional and it does not yet need a sophisticated UI

Version 2



view here: https://drive.google.com/file/d/1BQ5sAv1YHQ51OYqF1-ZDPdU5XOw5NSMA/view?usp=sharing

This version of the program is no longer as basic. It provides the following….

- A home screen
- Instructions
- Quit functionality
- Two different game board types
- One game board of each type
- A basic system for showing where the user is
- Random generating game boards
- Improved UI
- A progress bar
- Music capability
- User path is now highlighted
- User can now add a name

This is the second version of the program. It's purpose is to be functional and provide intuitive features and contain and a good UI

You may have more or less than 3 versions (these may be attached at the end of the report rather than screen shot here. If attached, still comment on any changes and additions).

Inputs:

What are your inputs: Example strName

- User name
- Sound on
- sound off
- click buttons/canvas
- keyboard entry

Outputs:

What are your outputs: Example strName

The outputs are the variables displayed on the screen. They are…

- Sound on/off
- current user position
- current user score (intTotal)
- Previous user Scores
- Winning Square
- UserName
- Starting Square

Name	Type	Global /local	Description/Scope
intMusic	integer	Global	Controls whether or no the music is playing
self.WinningTotal	integer	Global	Used within class to store the winning total. Can be edited outside of class
self.WinningSquare	integer	Global	Used within class to store the winning square.. Can be edited outside of class
intPlayed	integer	Global	Helps to control user level and music playback
label_image	string	Global	Holds the file path for the megaphone images
intInGame	integer	Global	Used to tells when the user is actually playing
intLevel	integer	Global	Used to tell what the users current level is
intGameMode	integer	Global	Used to tell if the user is playing a 3 X 3 or 4 X 4 grid
intTotal	integer	Global	Used to store the users total
intClicked	integer	Global	Used to store the square that the user clicked

intGradient	integer	Global	Used to determine the colour gradient of the next square
xLeft	integer	Global	used for colouring the squares
xRight	interger	Global	Used for Colouring the squares
yUp	integer	Global	used for colouring the squares
intSubGameMode (x2)	integer	Global	Used in two different definitions to control exit functionality
intPressd (x2)	integer	Global	Used in two different definitions to control exit functionality
intLeft	integer	local	used locally in several definitions to create the correct square
intRight	integer	local	used locally in several definitions to create the correct square
intBottom	integer	local	used locally in several definitions to create the correct square
intTop	integer	local	used locally in several definitions to create the correct square
intNumberCount	integer	local	used locally in several definitions to create the correct square
intRecreate(0, 1, 2, 3)	integer	global	There are four of these (intRecreate - intRecreate3). They control the messages shown when the user enters their name.
ment	stringVar	global	Used when the user enters their name.
strName	String	global	Used when the user enters their name. Prints it on all the screens
intLeave	integer	global	Used to control the popups when the user changes game mode
intNameEntered	integer	global	Prevents the user from having to re enter their name every time they visit the home screen
Message	string	global	The messages shown when the user enters a wrong name
Message1	string	global	The messages shown when the user enters a wrong name
mEntry	unknown	global	Variable that stores the entry box

Name	**Type**	**Description/Scope**
self.list	list	Used in the class to store the different board sequences
listUsed	list	Used to store the squares that the user has already clicked.
listTotal	list	Stores the users previous totals for easy backtracking
listUsedCoordinates	list	Stores the coordinates of the users last square
listMusicCoordinates	list	Used to store the needed location of the music button
listgradient	list	Stores all the possible colour gradients

Class(es), Objects, types and descriptions:

Name	Type	Information/objects Stored	Properties, Functions, Operations, Etc
class Grids:	Class	Grid layout, Winning Square, Winning Total	Holds the needed values of each grid
def __init__	Definition	self, list, WinningTotal, WinningSquare	Sets all the values in the class
def SetValues	Definition	list, WinningTotal, WinningSquare	Is used be other parts of the program to set the values within the class

Module Names, types and descriptions:

Name	Type	Description/Scope
Tkinter		Is imported at the beginning of the program and allows the a graphical user interface to be generated
exitButton	Object	A button that allows a user to exit the GUI/Application so they do not have to hard exit or complete the application to the end

Play button	Object	Allows the user to enter the play menu
Grid select button	Object	Allows the user to select the type of game that they wish to play
Instructions button	Object	Allows the user to view the instructions for the game
Music Button	Object	Allows the user to play/pause the music without have to adjust the volume on their computer.
return to home button	Object	Allows the user to return to home from any screen, thus negating the need for the user to quit the program and start again.
EntryBox	Object	Allows the user to type in their name
EntryButton	Object	Allows the user to pass their name into the program
HTP	Function/Definition	Generates an instructions screen so that the user can view how the game is played.
CopyRight	Function/Definition	Generates the copyright for each screen
ResetParameters	Function/Definition	Resets all the variables when the user goes to the home screen or changes games
GenerateMusic	Function/Definition	Creates the music button on each screen
PlayMusic	Function/Definition	Plays/stops the music when the music button is clicked
ExitCurrentGame	Function/Definition	Allows the user to switch game modes while in game
Press1	Function/Definition	Used by the program to determine the users current game mode when switching games
Press2	Function/Definition	Used by the program to determine the users current game mode when switching games
Play	Function/Definition	Generates a screen of further options so that the user can decide which game mode to play
GenerateProgressBar	Function/Definition	Generates and updates the progress bar when he user levels up
GenerateGrid	Function/Definition	Fills the grid based on the user's game mode
GenerateReturnToHome	Function/Definition	Generates the return to home button for each screen
Quit	Function/Definition	Gives the program quit functionality and asks the user if they really want to quit

CreateHomeButtons	Function/Definition	Generates the buttons on the users home screen and allows the user to continue through the game
DeleteMessage	Function/Definition	Deletes the error message shown if the user enters an invalid name
DeleteMessage1	Function/Definition	Deletes the error message shown if the user enters an invalid name
limitSize	Function/Definition	Limits the size of the name that the user can input
NameCheck	Function/Definition	Checks that the user name meets the specified criteria
GenerateHome	Function/Definition	Generates the home screen so that the same code does not need to be repeated
ColourGradient	Function/Definition	Changes the colour of the user location to make backtracking easier
ThreeByThree	Function/Definition	Genterates and allows the user to play a 3 X 3 board.
FourByFour	Function/Definition	Genterates and allows the user to play a 4 X 4 board.
LevelUp	Function/Definition	Allows the user to level up and tracks the level
PopupDestroy	Function/Definition	Destroys pop ups in the game allowing the user to continue
GenerateTile	Function/Definition	Generates the coloured tile when the user clicks a square
GenerartBoard	Function/Definition	Randomly generates each game board allowing for greater game flexibility
callback(event)	Function/Definition	Checks where the user has clicked and performs appropriate actions
__init__	Function/Definition	Adds the values from the randomly generated board and uses them to generate the grid
SetValues	Function/Definition	Sets the values for the boards

Once your plan and flowchart ()Modular structure) are complete, develop a comprehensive test plan to use to test your program. Add more tests as necessary

Test Plan.				
Test Type	**Item tested**	**Test inputs?**	**Expected outcome**	**Comments**
Expected[#]	GenerateTile	Click on valid square	The quare should change to a shade of blue	NA
Expected[#]	Music	The user clicks on the music button when the button is a white megaphone	The music should start playing and the megaphone should turn red	NA
Expected[#]	Music	The user clicks on the music button when the button is a red megaphone	The music should stop playing and the megaphone should turn white	NA
Expected[#]	RTH	When in module RTH. The user clicks H, h, 1, 2	The program should regenerate the homescreen.	NA
Expected	mEntry	The user enters an alphanumeric name between 2 and 17 characters	The program should accept the name a generate the homescreen	NA
Boundary[+]	GenerateMusic	The user clicks on the on the boundary of the button (megaphone button)	If the user clicks within approximately 2-3px of the megaphone icon it should work. Otherwise it will do nothing, inside the 2-3px it will work.	NA
Boundary[+]	mEntry	The user enters a name that is 16, 17 and 18 characters	The Program should accept 16 and 17 and will show an error for 18.	NA
Boundary[+]	mEntry	The user enters a name that is 1, 2, 3 characters.	The program should accept the 2 and 3 and show an error for the 1	NA

Invalid	mEntry	The user enters a name that is greater than 17 characters	The program should display an error message and disallow any further input	NA
Invalid	mEntry	The user enters a name that is less than 2 characters	The program should display an error	NA
Invalid *	mEntry	The User enters Numbers instead of letters	The program should display an error message	NA
Invalid *	Callback Event	The user clicks a square that is not adjacent. (used or unused)	The program should not do anything	NA
Invalid *	Callback Event	The user clicks outside the gameboard	The program should not do anything	NA
Invalid	mEntry	The user enters a series of symbols such as !@#$%^&*	The program should not accept the name	NA
Invalid	Callback Event	The user clicks on the start square	The program should do nothing	NA

*#Achieved +Merit *Excellent*

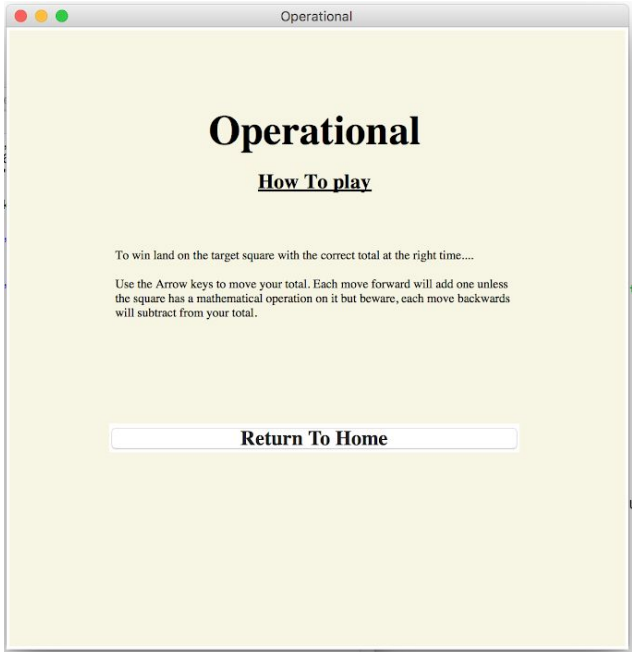
Part 2 - Producing the program

You may have more than 3 versions (these may be attached at the end of the report rather than screen shot here. If attached, still comment below on any changes and additions

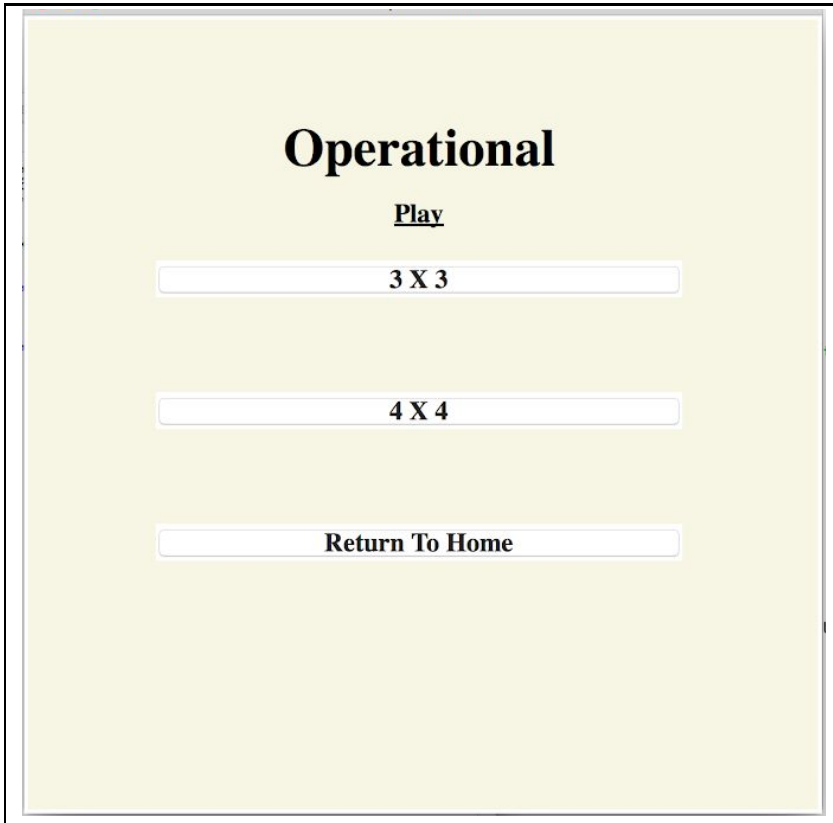
Note that the music button does not appear because it is disabled on MAC OS and I am using a macbook to test. There is also a colour difference in the background in

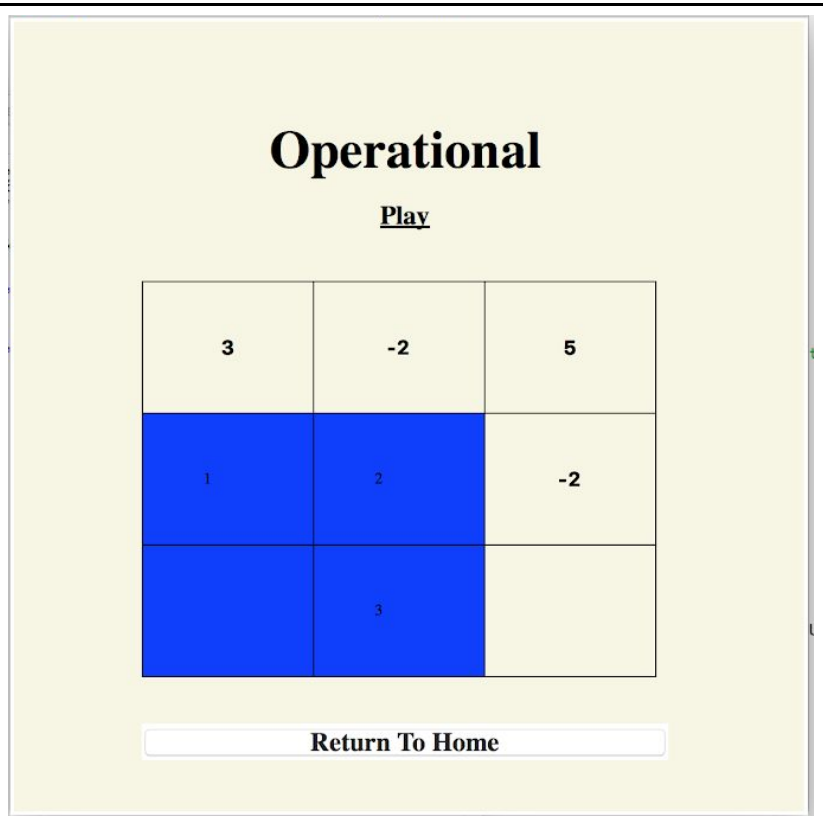
some screenshots. This is due to a colour change that I have made for Aesthetic reasons in later versions of the program. All program fundamentals remain exactly the same.

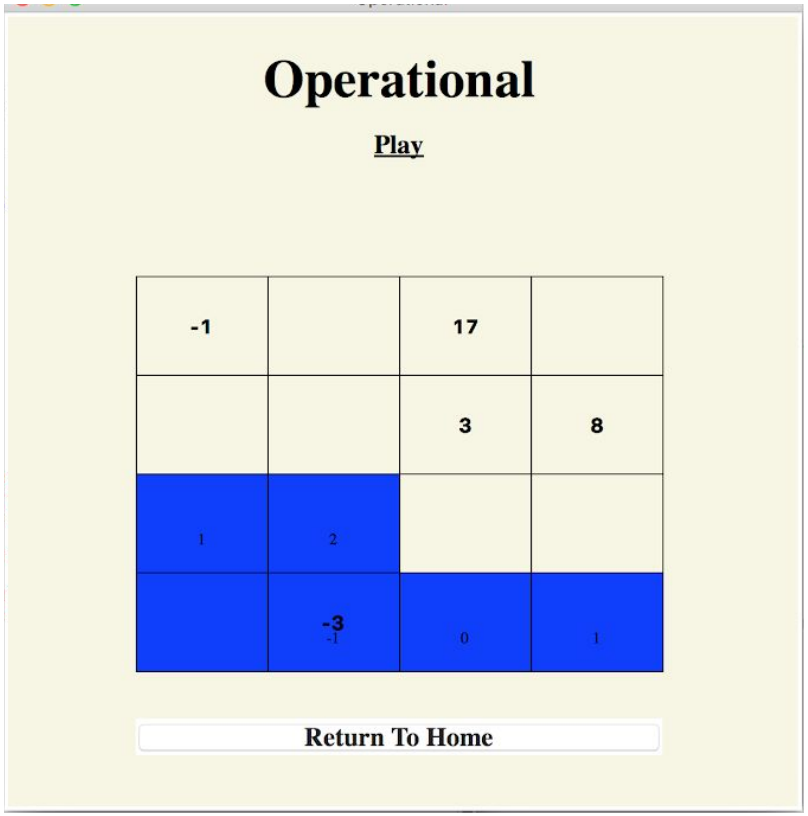
Program version(s):
Version 1



*Lable **FirstInitial, Lastname, underscore, version (GHall_v1)** and **Place in drop box***


Operational
Play
3 X 3
4 X 4
Return To Home


Operational
Play
3
-2
5
1
2
-2
3
Return To Home

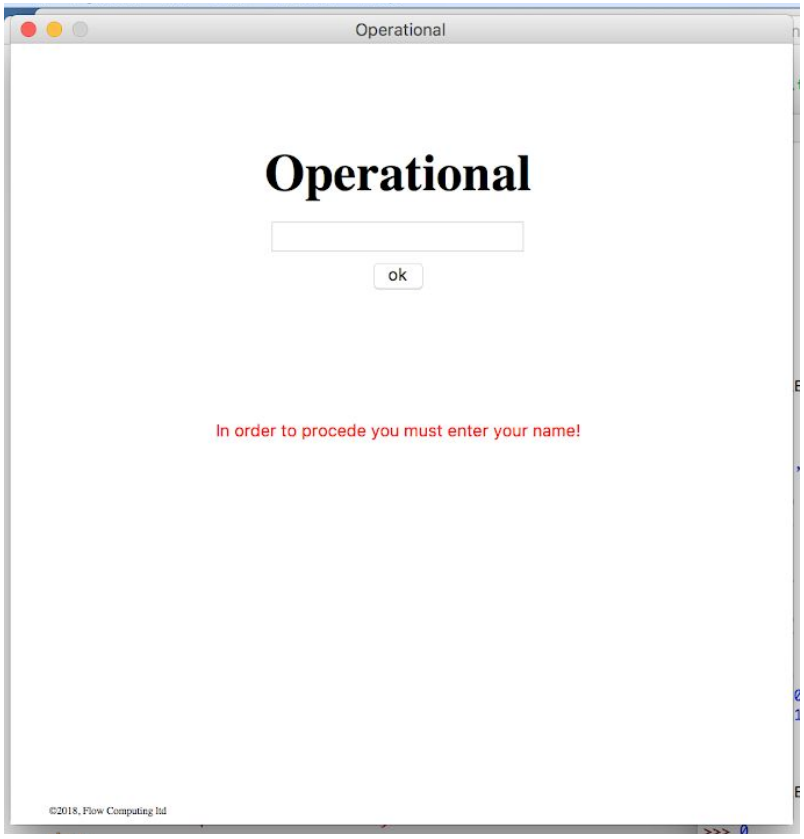


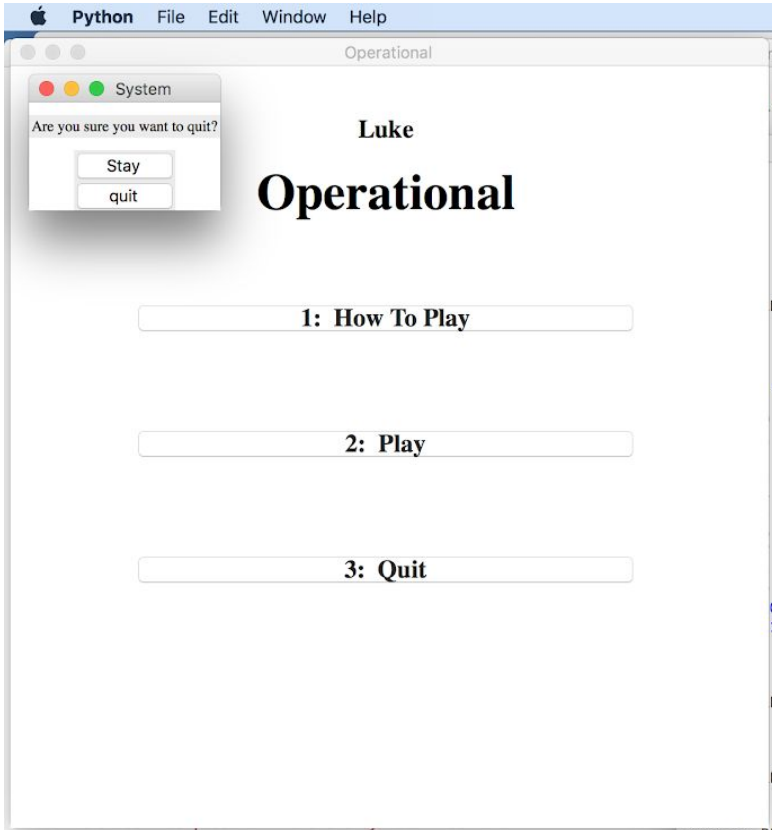
Features:

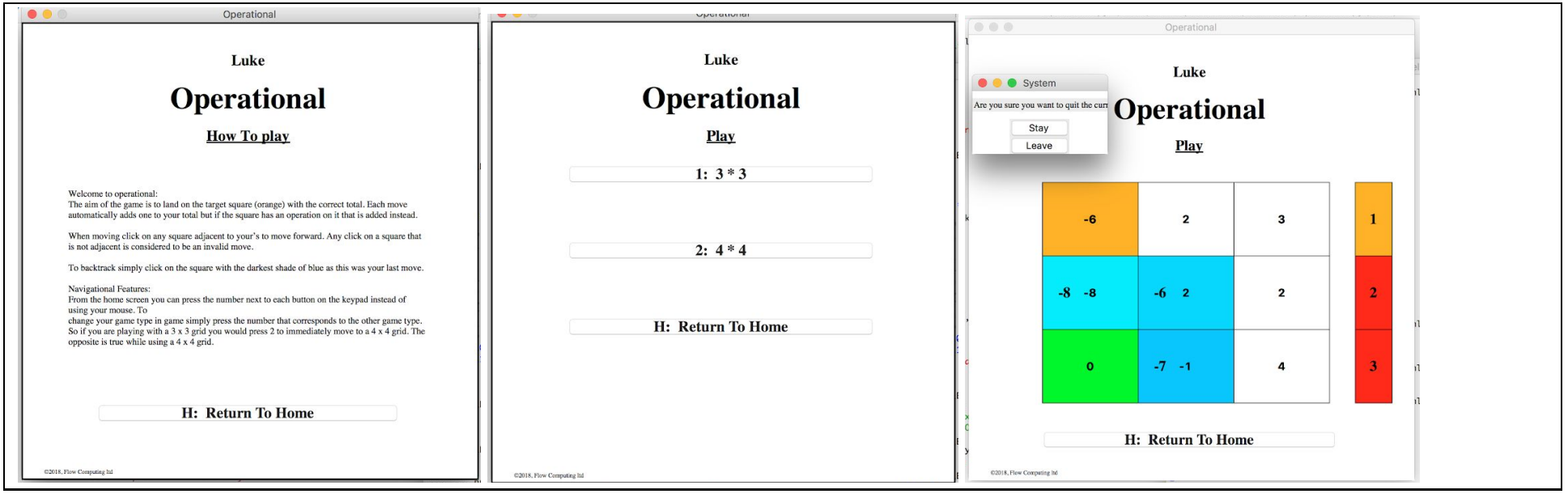
- A home screen
- Instructions
- Quit functionality
- Two different game board types
- One game board of each type
- A basic system for showing where the user is

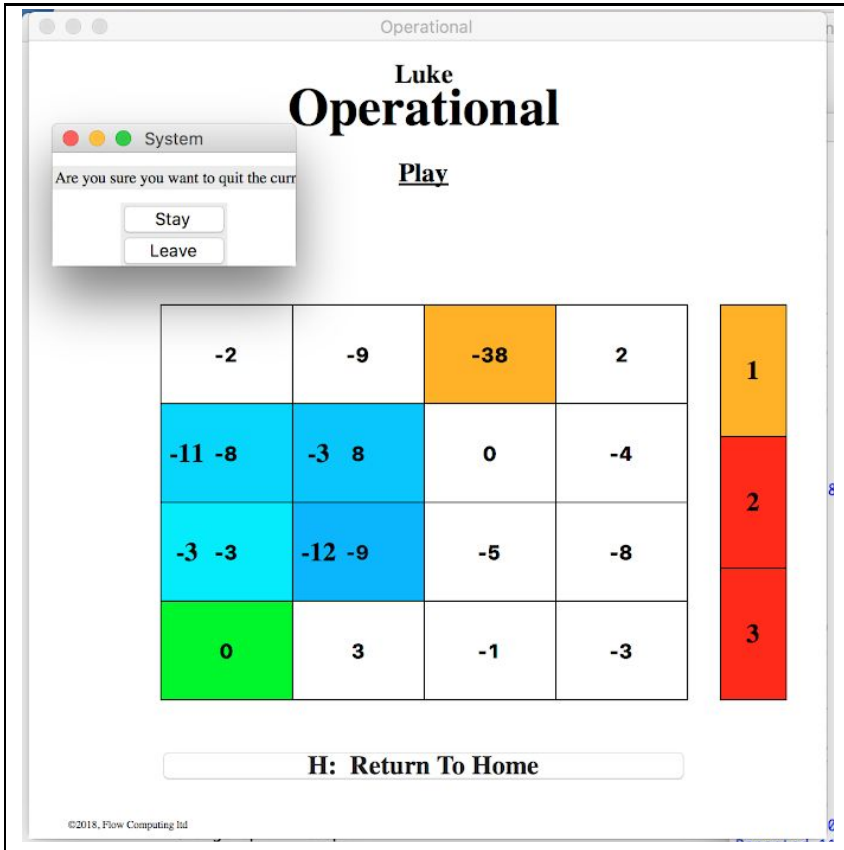
Version 2

*Lable **FirstInitial, Lastname, underscore, version (GHall_v2)** and **Place in drop box***






Operational
Luke
Operational
How To play
Welcome to operational:
The aim of the game is to land on the target square (orange) with the correct total. Each move automatically adds one to your total but if the square has an operation on it that is added instead.
When moving click on any square adjacent to your's to move forward. Any click on a square that is not adjacent is considered to be an invalid move.
To backtrack simply click on the square with the darkest shade of blue as this was your last move.
Navigational Features:
From the home screen you can press the number next to each button on the keypad instead of using your mouse. To
change your game type in game simply press the number that corresponds to the other game type. So if you are playing with a 3 x 3 grid you would press 2 to immediately move to a 4 x 4 grid. The opposite is true while using a 4 x 4 grid.
H: Return To Home
©2018, Flow Computing ltd
Luke
Operational
Play
1: 3 * 3
2: 4 * 4
H: Return To Home
©2018, Flow Computing ltd
Operational
System
Are you sure you want to quit the curr
Stay
Leave
Luke
Operational
Play
-6
2
3
1
-8 -8
-6 2
2
2
0
-7 -1
4
3
H: Return To Home
©2018, Flow Computing ltd



Description of what has changed/is new and why?

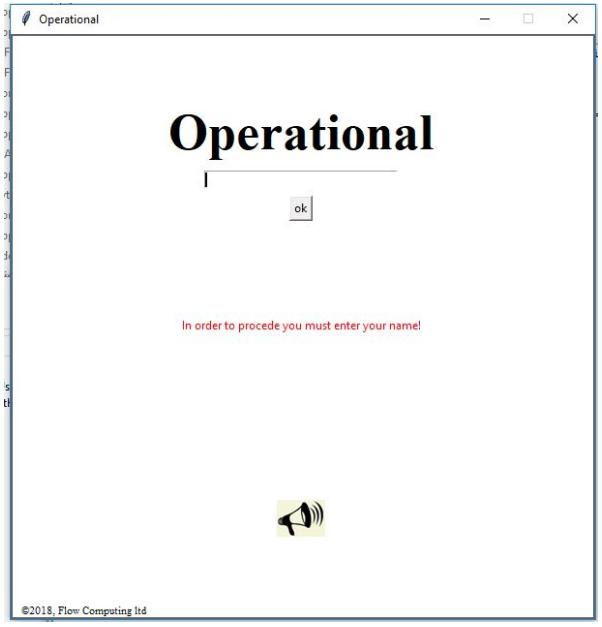
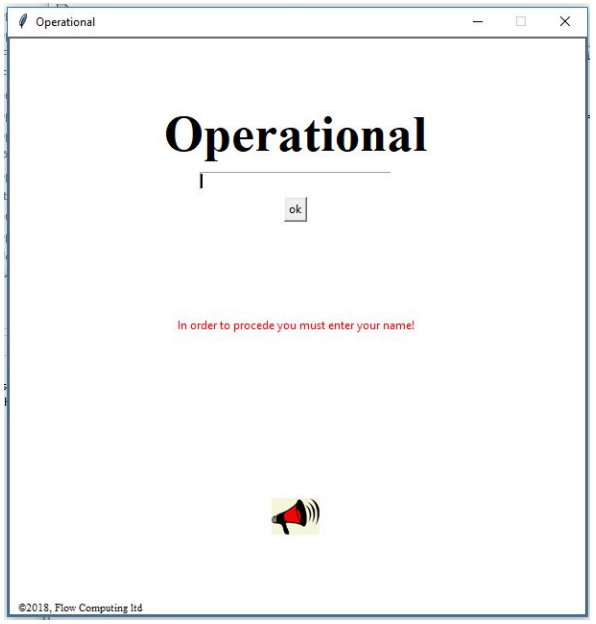
- Updated Instructions to make gameplay easier
- Two different game board types, randomly generated to improve user experience
- Improved user position system to make gameplay easier
- Music capability to aid user enjoyability
- Improved UI to make the user experience better
- A progress bar to aid the user experience by adding a sense of purpose
- The ability for the user to enter their name giving a sense of personalisation

Results

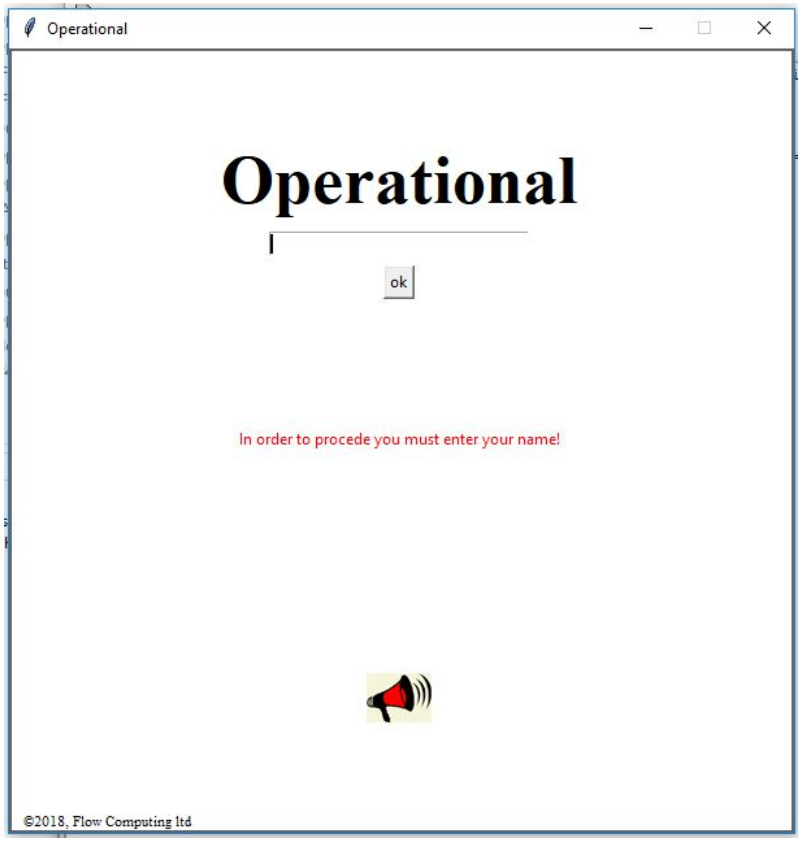
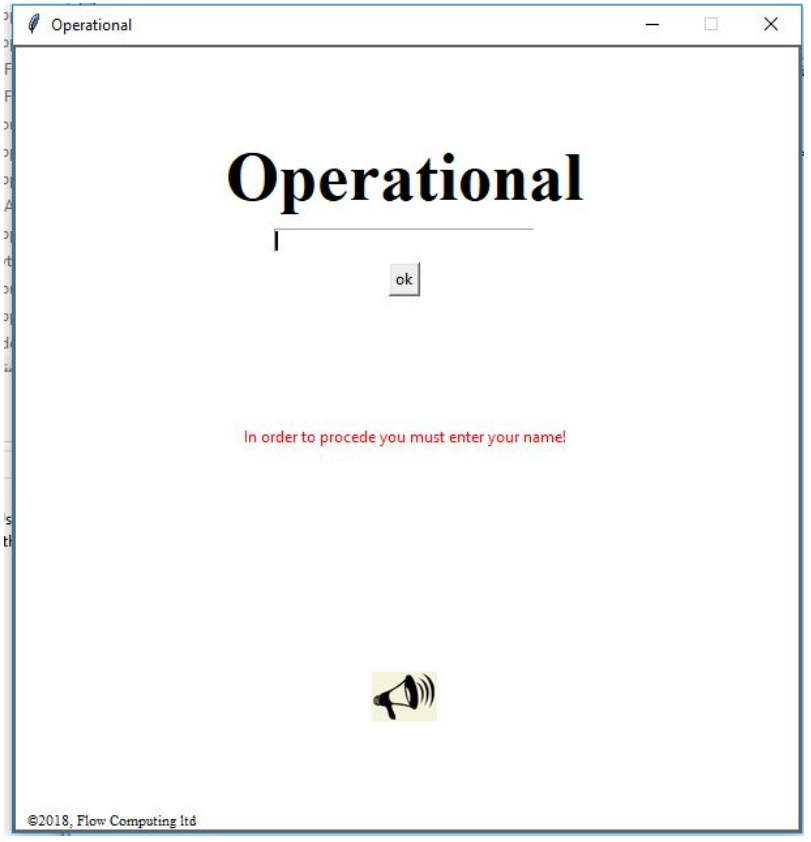
Collect and comment evidence (screenshots), documenting how you followed your planned testing procedures. Include screenshots of any modifications you made to your program as a result of the testing.

Comment screenshots of completed tests ***(4A; 7M; 10E):***

Marker please note that the background colour is different in some screenshots. This is due to an Aesthetic change made later on. The colour of the music button remained because I had not yet been able to change it due to my lack of PhotoShop at home

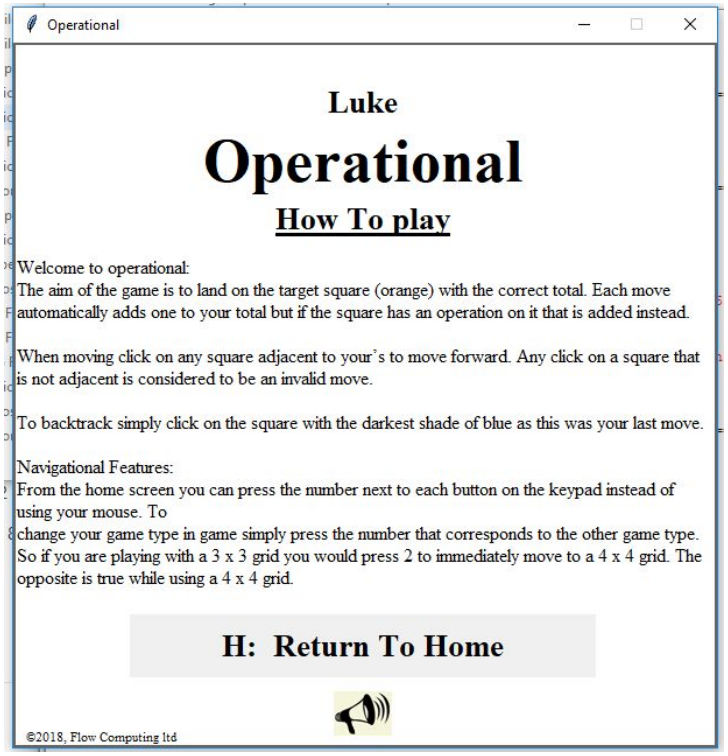
Expected: User clicks in centre of music button when white	
Before screenshot 	After screenshot 

After Improvement screenshot (if needed)
Notes: Program worked first time. No changes were needed

Expected: User CLicks on music button ehrn the button is a white megaphone	
Before screenshot	After screenshot
	
After Improvement screenshot (if needed)	
Notes: Program worked as needed first time	

Expected: RTH user clicks on h, H, 1 or 2

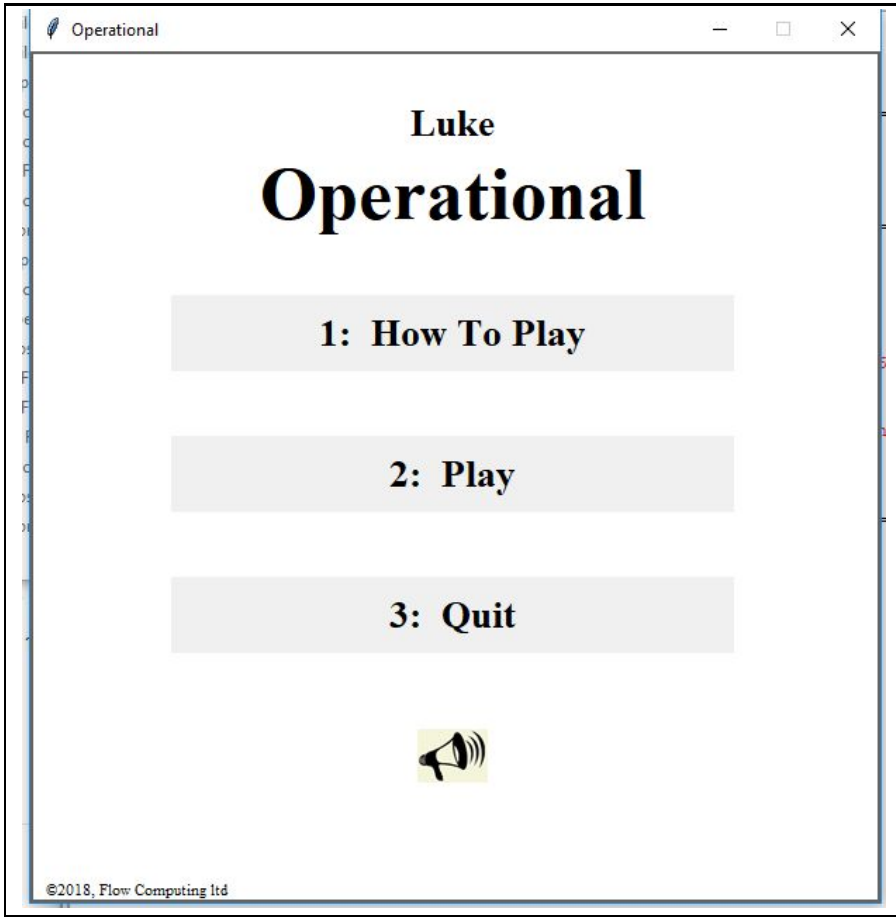
Before screenshot



After screenshot

```
Exception in Tkinter callback
Traceback (most recent call last):
  File "C:\Users\Daniel Hawinkels\AppData\Local\Programs\Python\Python36-32\lib\
tkinter\__init__.py", line 1699, in __call__
    return self.func(*args)
  File "C:\Users\Daniel Hawinkels\Downloads\Operational.py", line 12, in HTP
    root.bind('h', GenerateHomes)
NameError: name 'GenerateHomes' is not defined
```

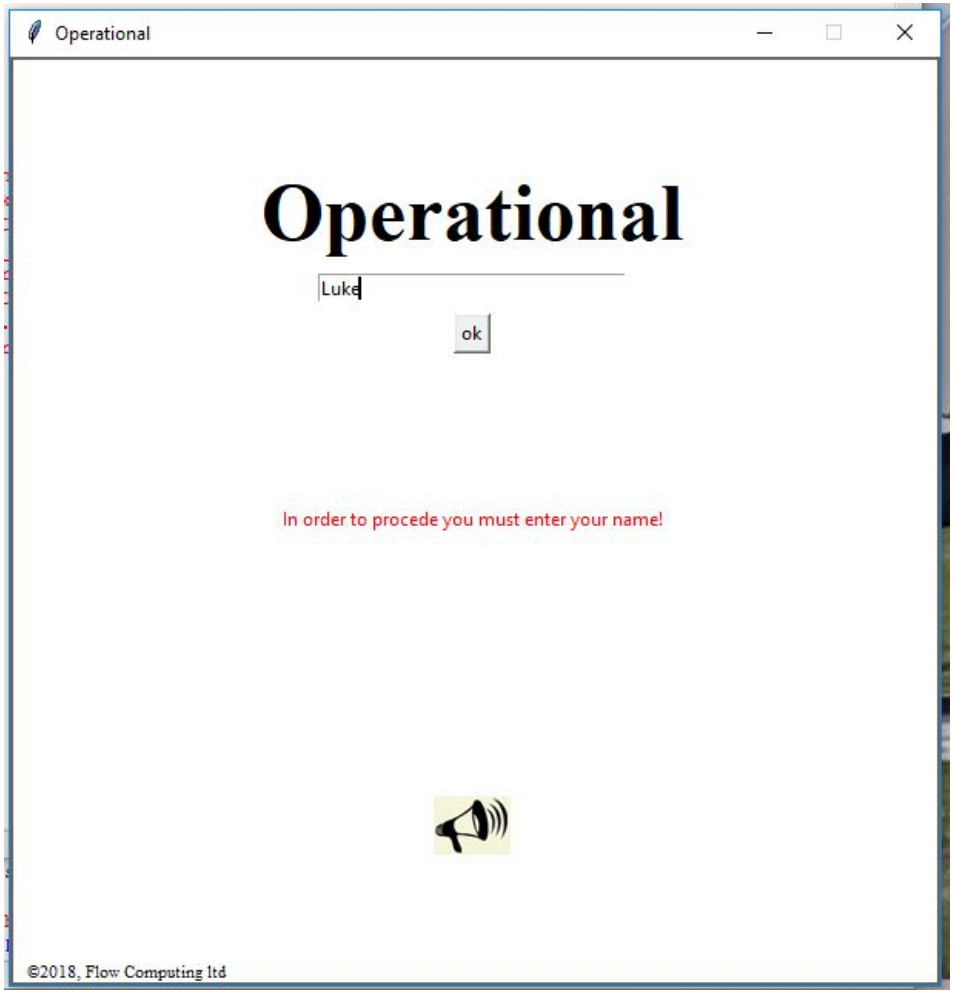
After Improvement screenshot (if needed)



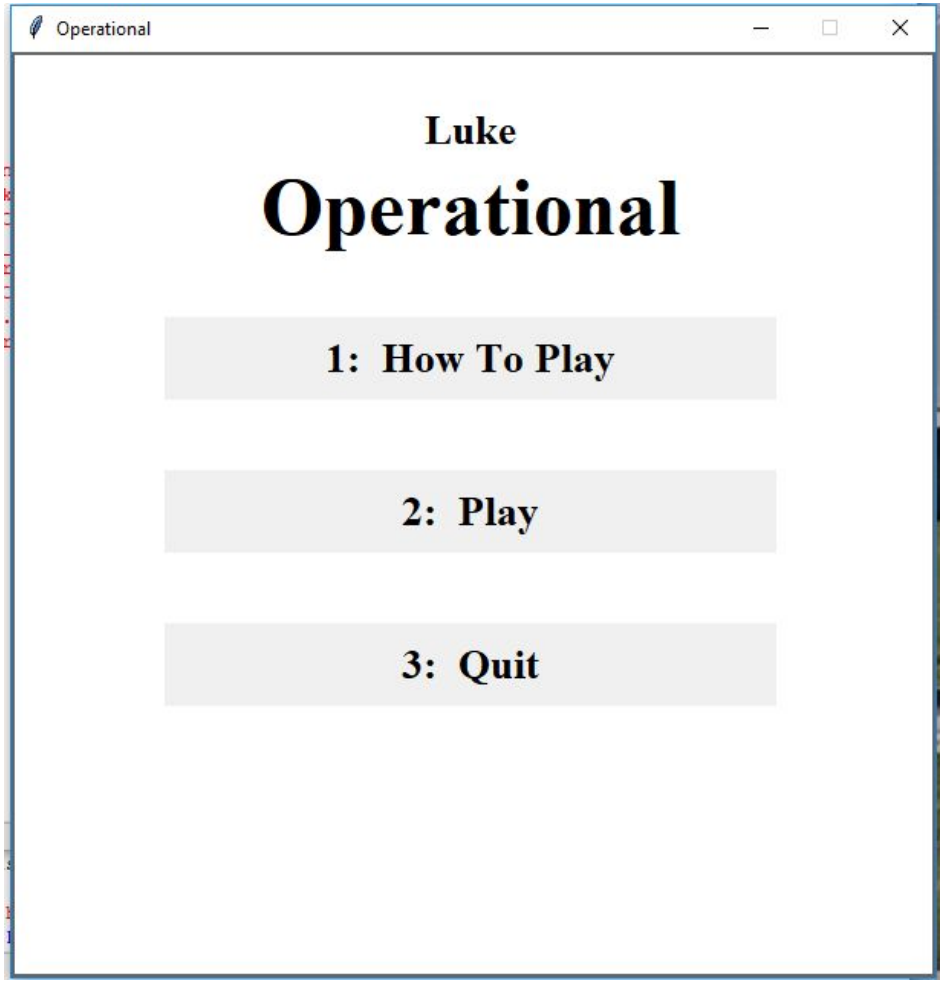
Notes: The error was due to a typo when calling the generate home function. This was fixed and the program worked fine

Expected: mEntry The user enters between 2 and 17 characters

Before screenshot

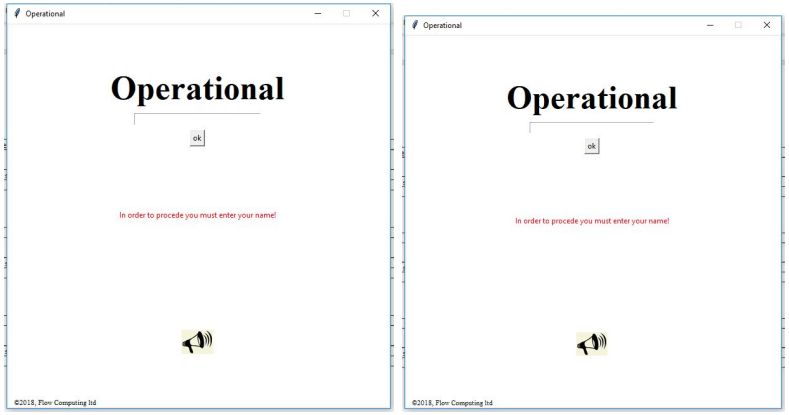
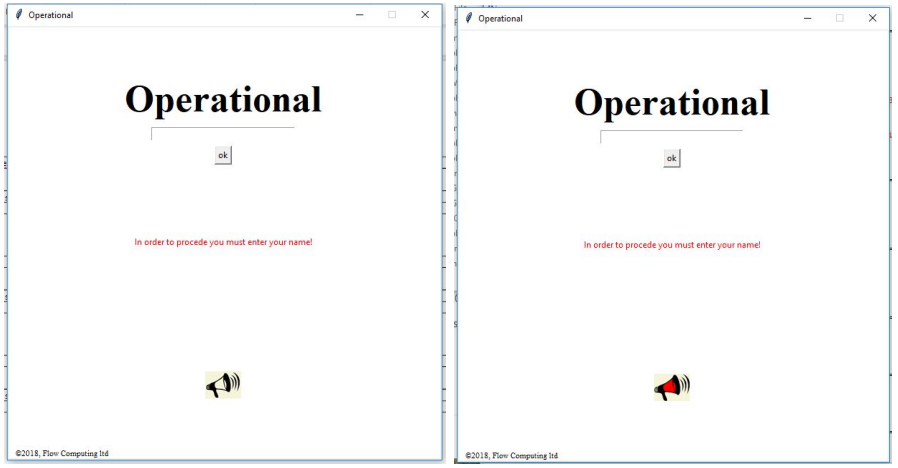


After screenshot



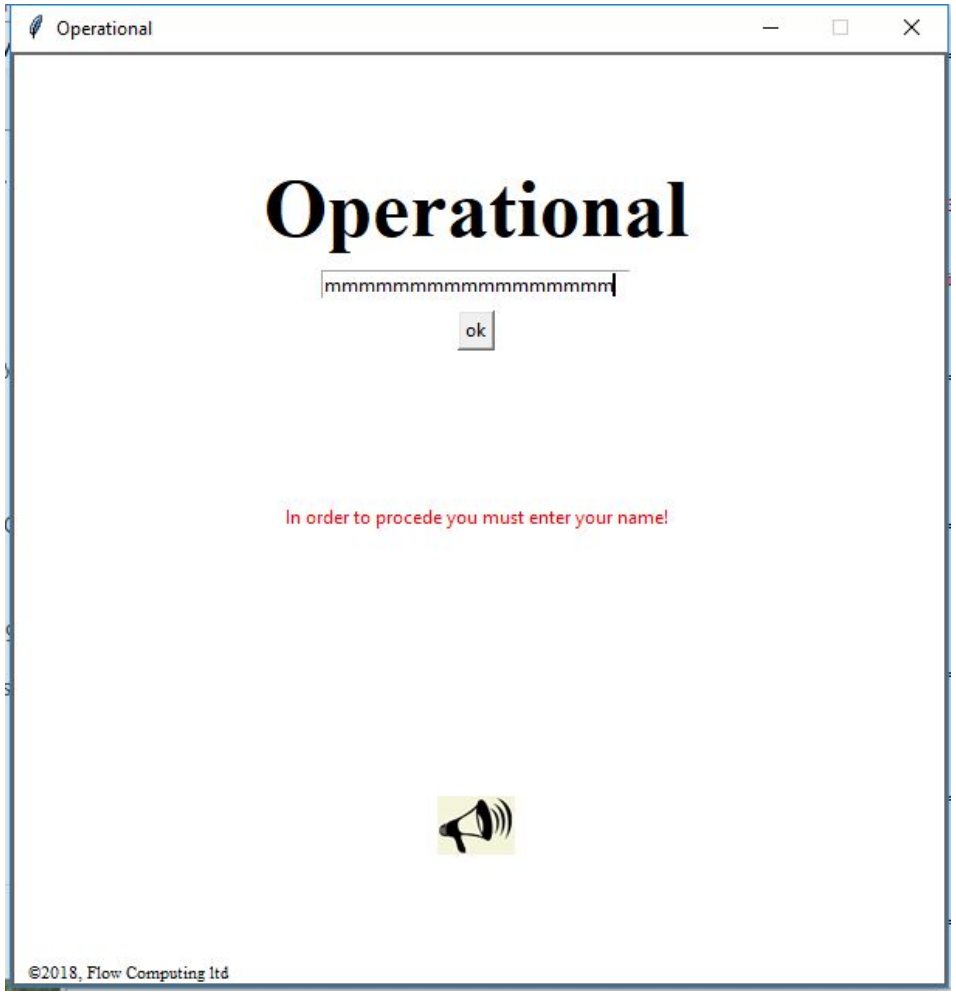
After Improvement screenshot (if needed)

Notes: Program worked fine first time

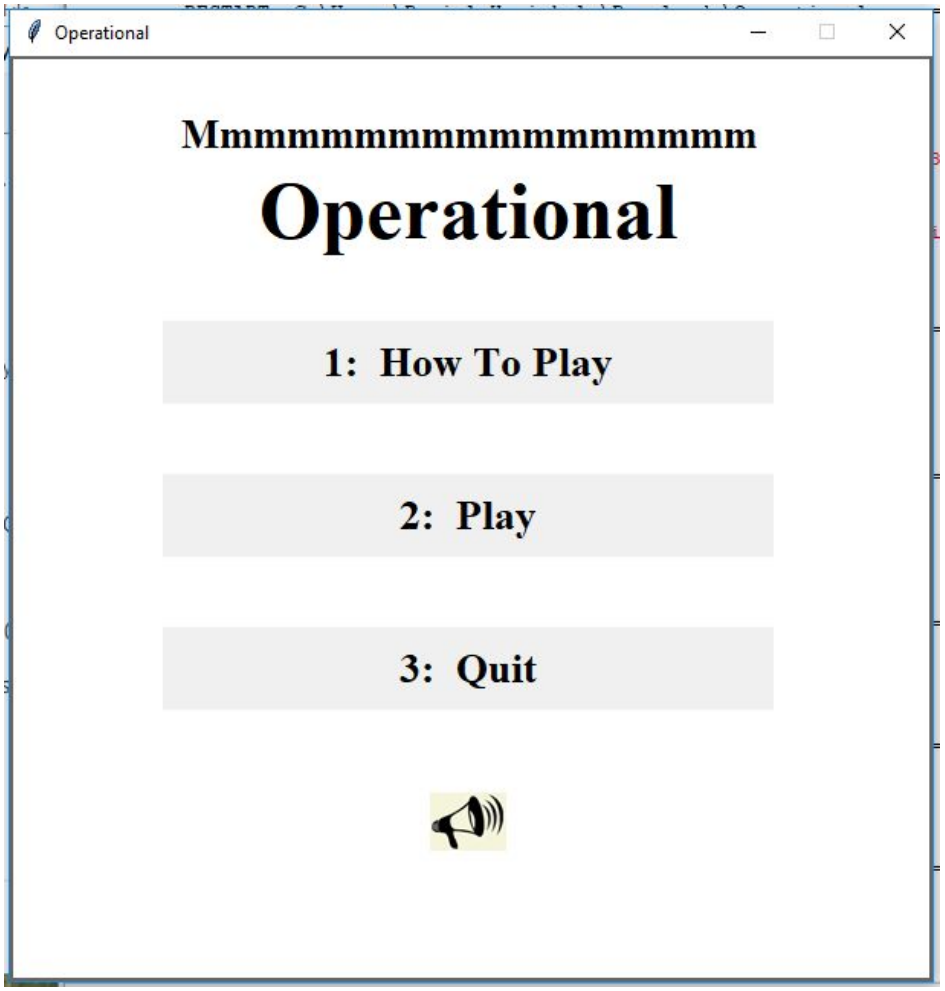
Boundary: GenerateMusic	
Before screenshot 	After screenshot 
After Improvement screenshot (if needed)	
Notes: Worked fine first time. **Note that fist 2 screenshots are clicking approx 4-5px away, 2nd 2 are 2-3px away.**	

Boundary: User enters a Name that is 17 Characters, I used 17 M's because these are the largest characters

Before screenshot

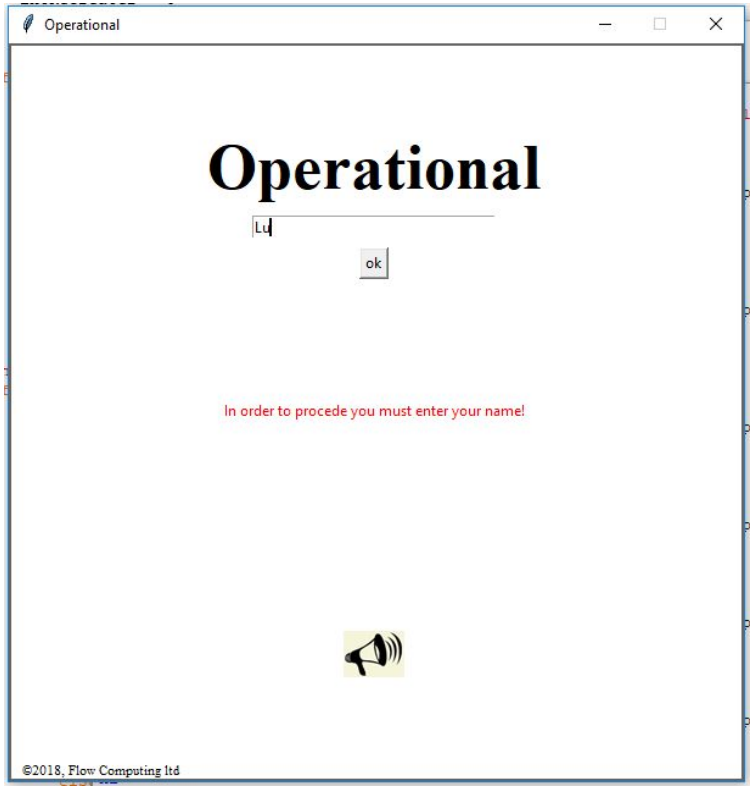
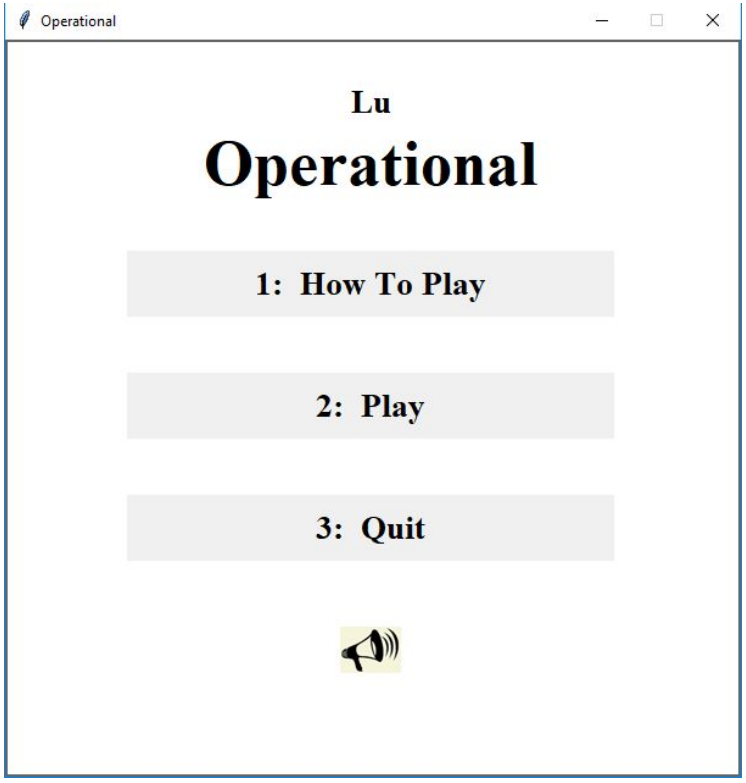


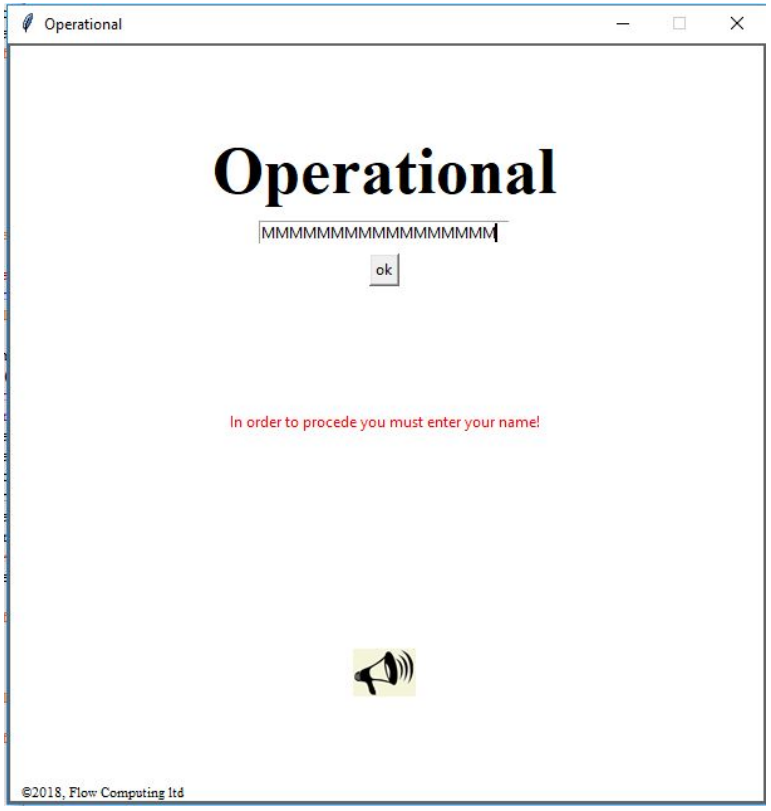
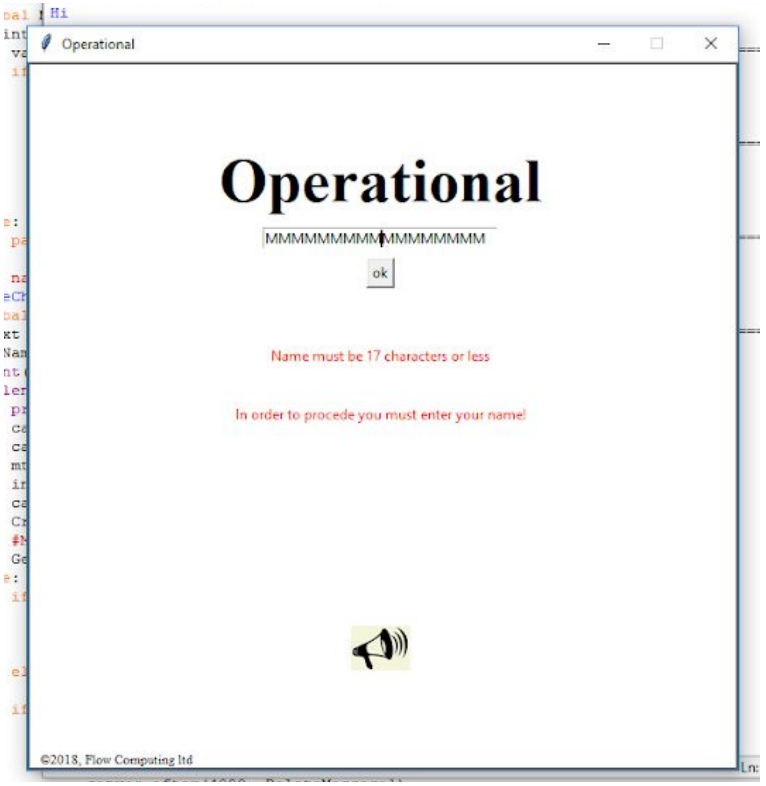
After screenshot



After Improvement screenshot (if needed)

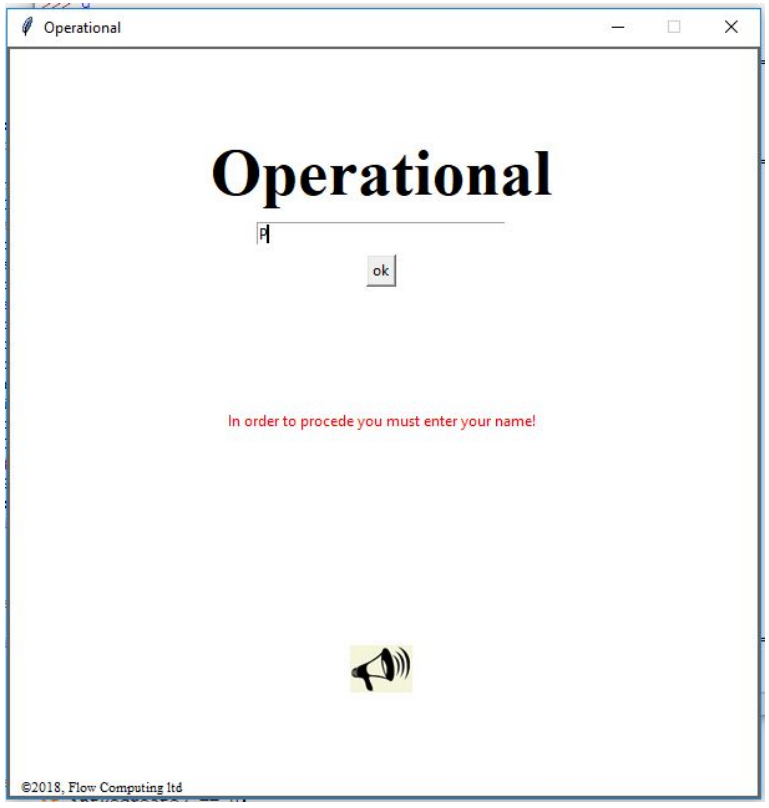
Notes: Worked fine first time

Boundary: user enters name that is 2 characters long	
Before screenshot  	After screenshot 
After Improvement screenshot (if needed)	
Notes: Program worked fine first time	

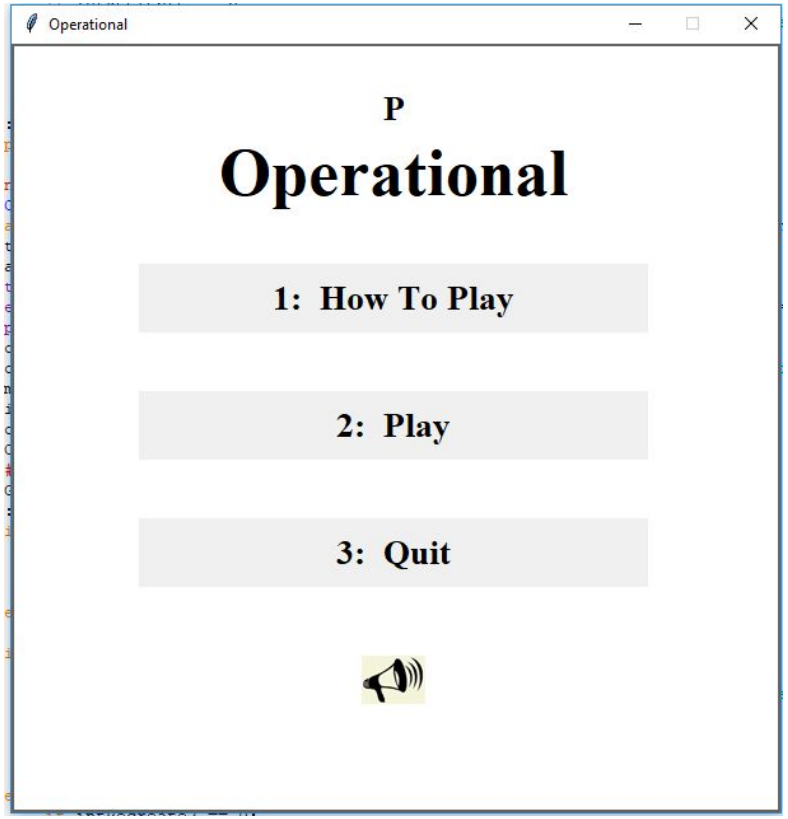
Invalid: User enters name that is greater than 17 characters.	
Before screenshot  	After screenshot 
After Improvement screenshot (if needed)	
Notes:The first few times the box was not big enough. After correcting it worked fine	

Invalid: The User enters a name that is less than 2 characters

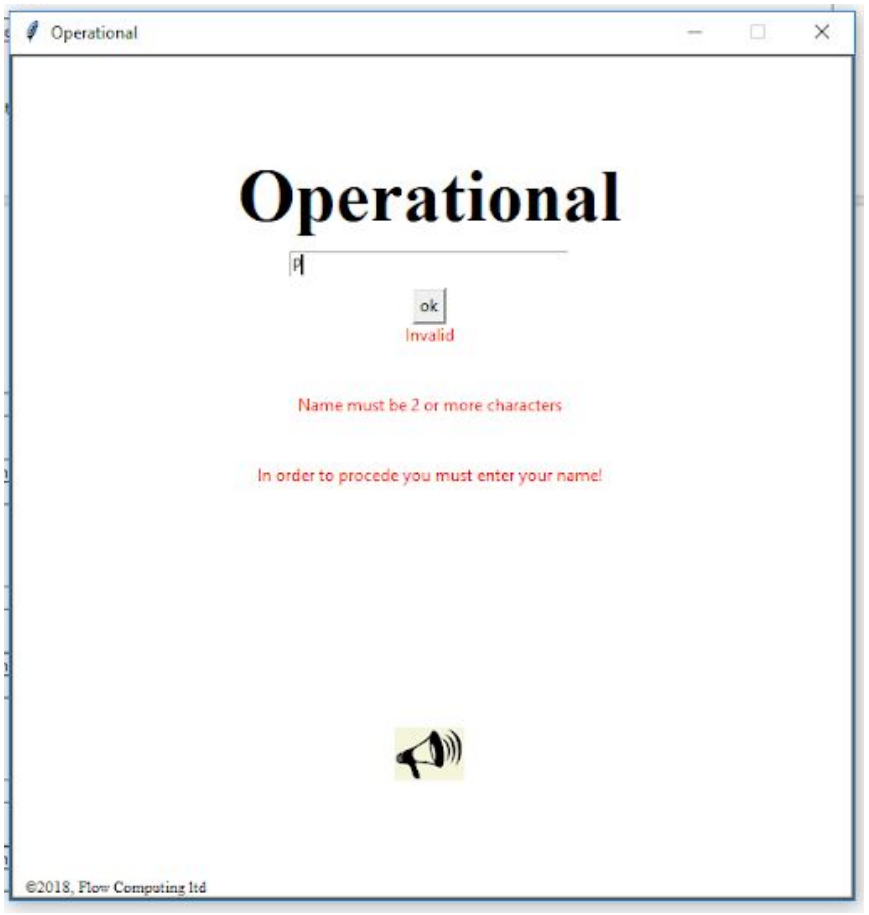
Before screenshot



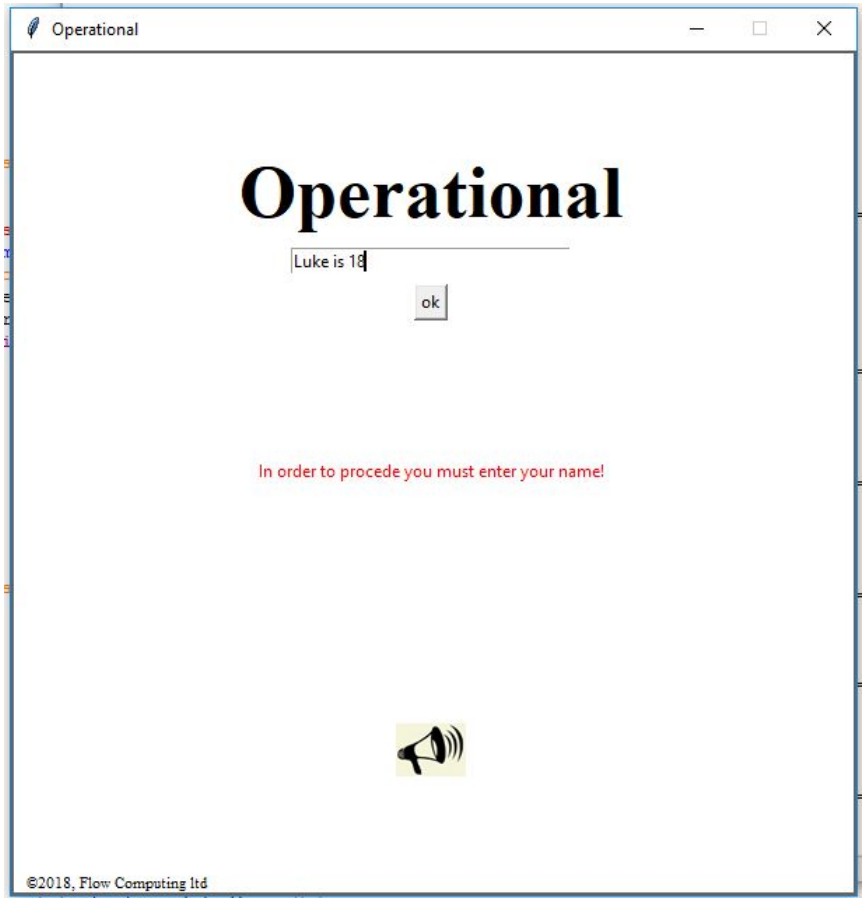
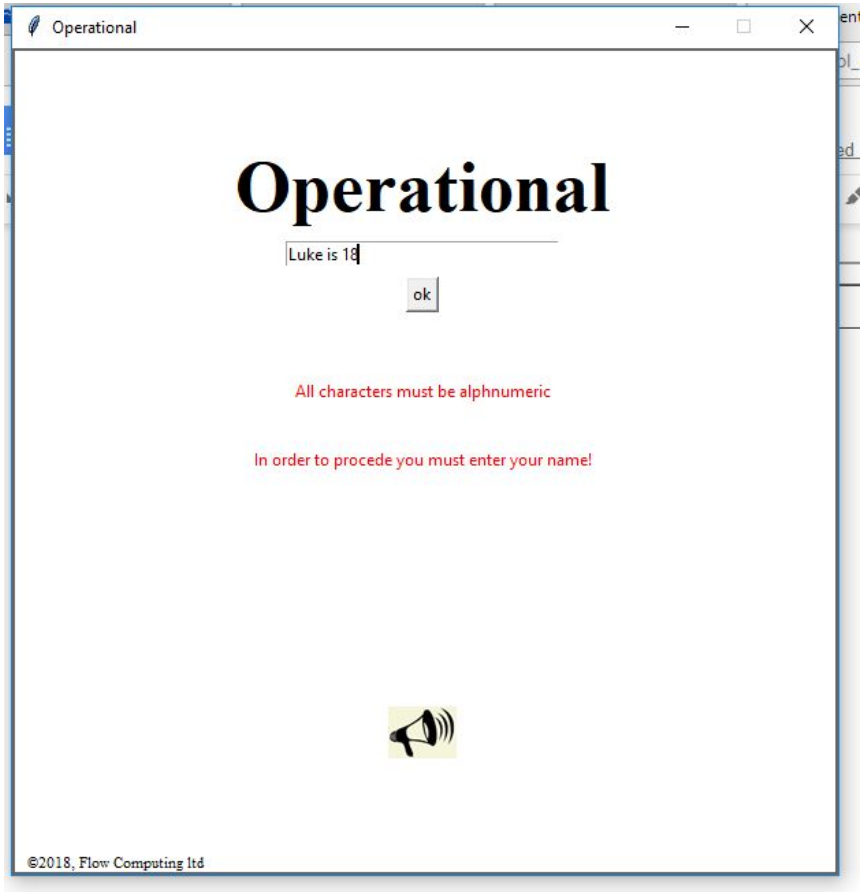
After screenshot

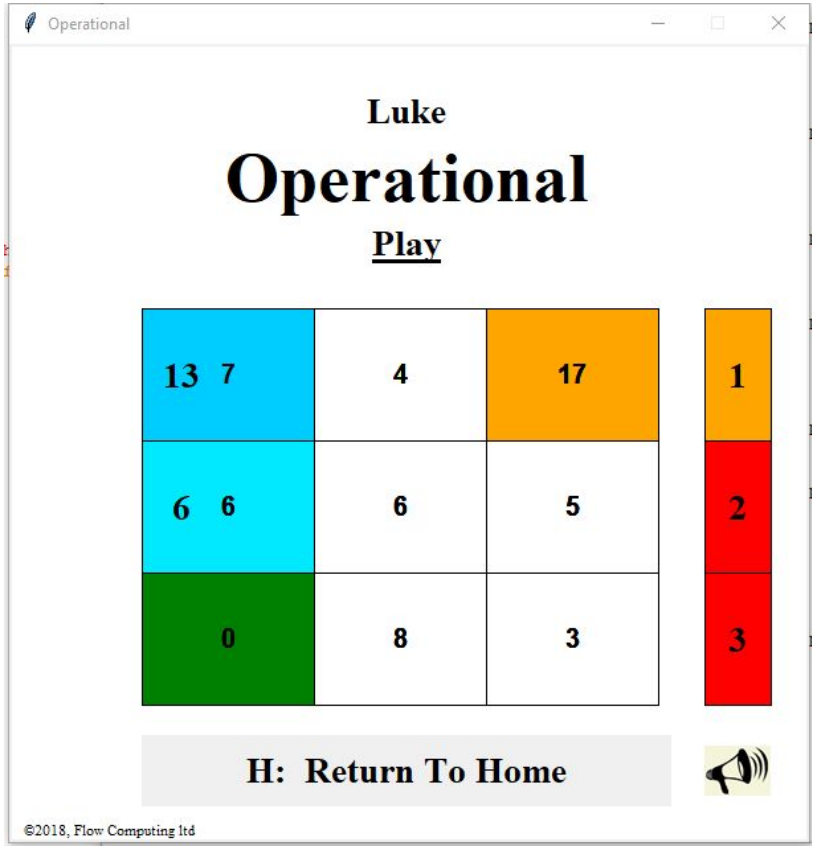
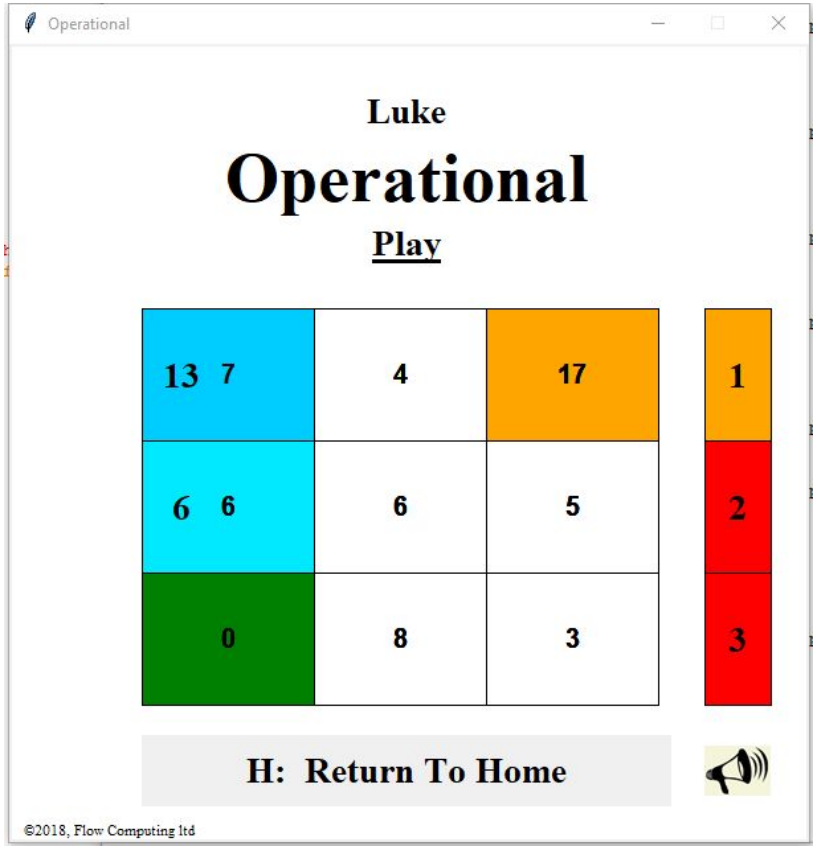


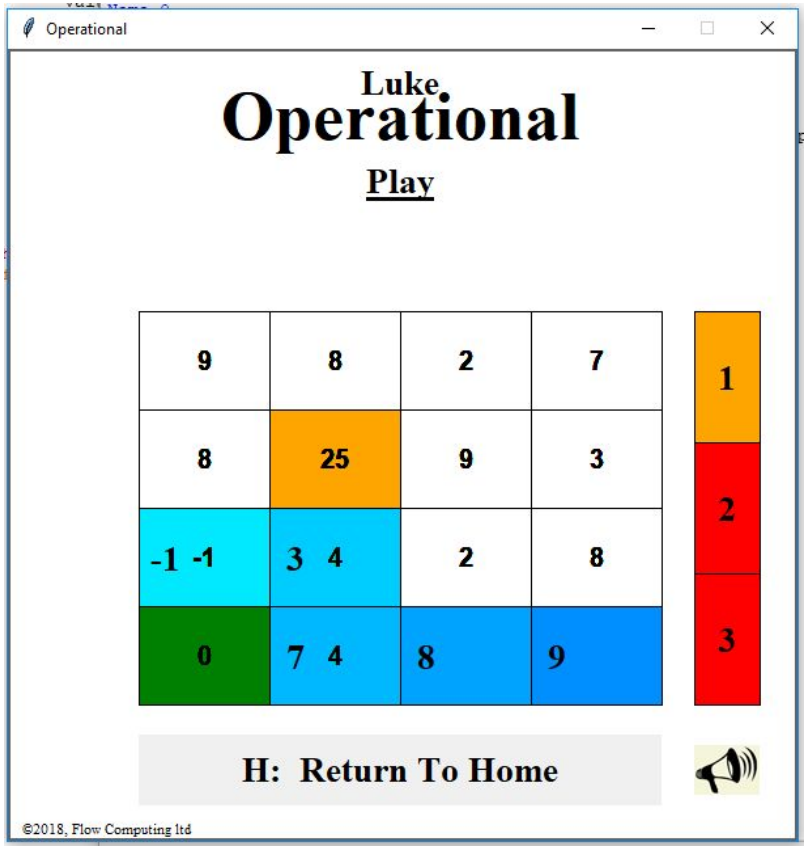
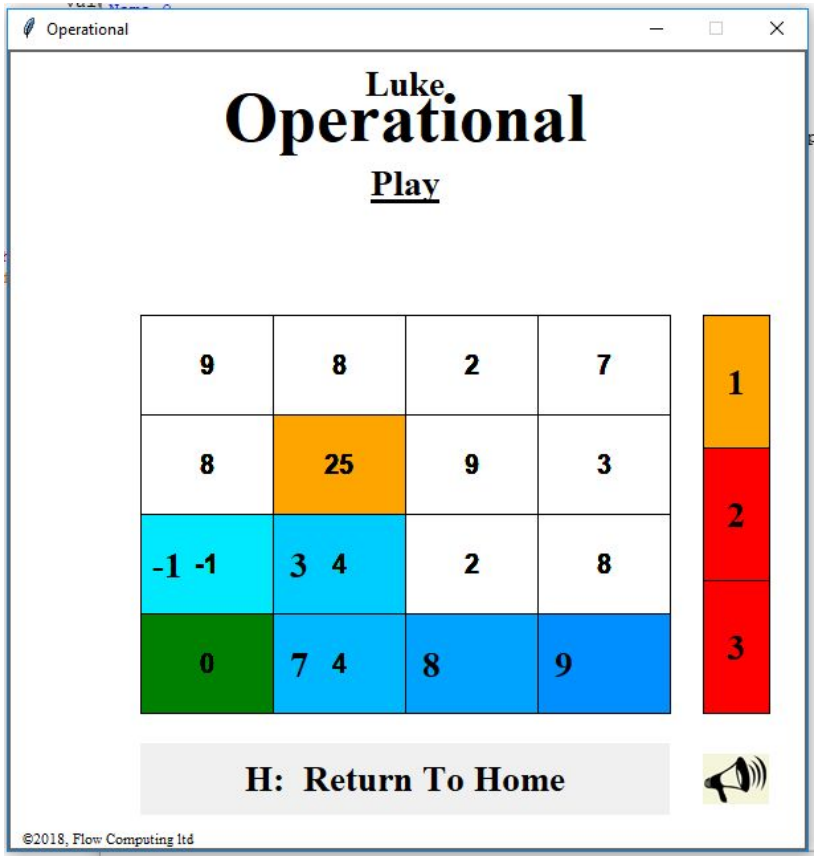
After Improvement screenshot (if needed)



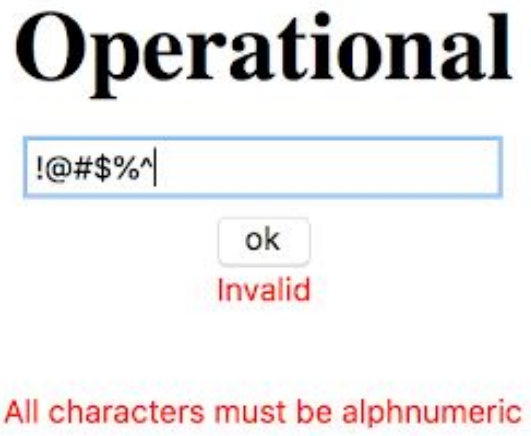
Notes: Due to a mistake in my code the program allowed for an input of 1 character

Invalid: the user enters numbers instead of letters	
Before screenshot	After screenshot
	
After Improvement screenshot (if needed)	
Notes: Worked fine the first time	

Invalid: The user clicks on a square that is not adjacent (can be used or unused)	
Before screenshot	After screenshot
	
After Improvement screenshot (if needed)	
Notes: The program worked first time. **I used the square marked 5 for this test.**	

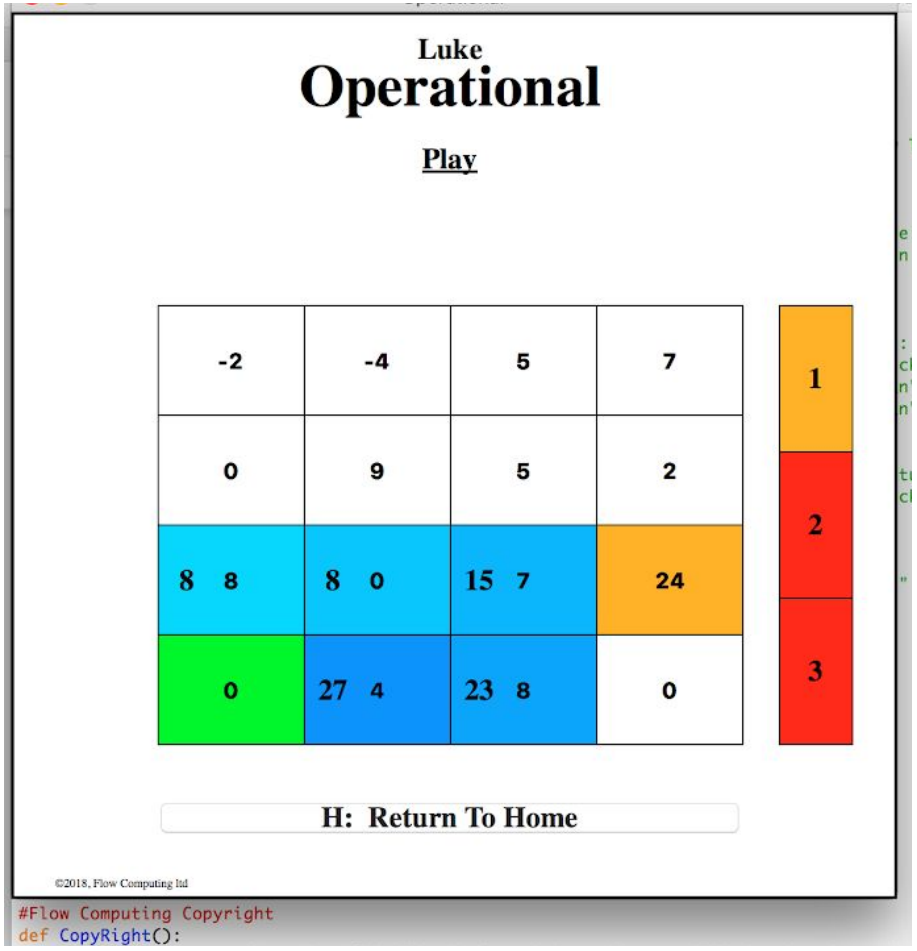
Invalid: The User clicks outside the gameboard	
Before screenshot	After screenshot
	
After Improvement screenshot (if needed)	
Notes: The program worked first time. **I used the area around the flow computing copyright for this test.**	

Add more tests as necessary.

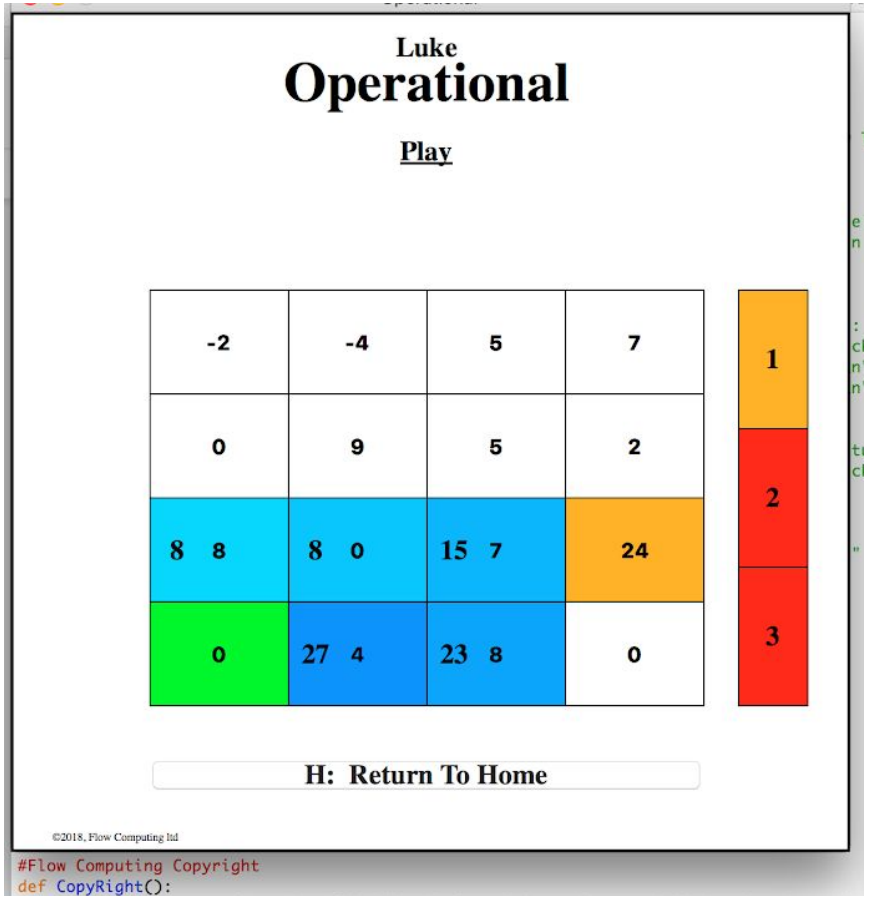
<table>
<tr><td colspan="2">Invalid: The User enters symbols such as !@#$%</td></tr>
<tr><td>Before screenshot
 
</td><td>After screenshot
 
</td></tr>
<tr><td colspan="2">After Improvement screenshot (if needed)</td></tr>
<tr><td colspan="2">Notes: The program worked first time.</td></tr>
</table>

Invalid: The User clicks the starting square

Before screenshot



After screenshot



After Improvement screenshot (if needed)

Notes: The program worked first time.