To set up the game, first download the zip file from the jjamm github and move it into whatever directory you prefer. Unzip the file. From this point forward, everything must be done using a linux command line app such as Terminal on mac or Putty. Start by entering the directory (for example, if you have the directory on your desktop and haven't changed the name, the command should be something like

```
cd Desktop/Level-JJAMM-main
```

To play the game, simply type

```
./dungeonjjamm
```

You will be prompted with four options. 'Start', 'Tutorial', 'Load Custom', and 'Quit.' Pressing the up and down arrow keys will highlight the different options, and pressing enter will choose one.
Choosing 'Start' will play the included level pack.
Choosing 'Tutorial' will play an included pack of tutorial levels, which will teach you about the mechanics of the game. It is suggested that you start with this option.
Choosing 'Load Custom' will allow you to play a custom level pack designed by yourself. This will be explained later in the document.
Choosing 'Quit' will end the program.

Another program is included in the download which allows you to create your own maps. Before running this, it is suggested that you create your own directory in which to house the levels, unless you would like to make changes to the included level packs. To do this, from the folder containing the dungeonjjamm program, type

```
cd AllLevels
```

Then make your own directory, naming it whatever you wish. For example, to create a directory titled 'myLevels' type

```
mkdir myLevels
```

Then, return to the previous directory. From here, you can run the mapCreator program by typing

```
./mapCreator
```

Upon running, you will first be instructed to enter the directory from which you will be editing or adding files. Using the above example, you would type

```
myLevels
```

You will then be prompted to enter the name of the file you wish to edit or add. If you type the name of an existing file, that file will be loaded and you will be able to edit it. Beware though, if your terminal size is too small to be displayed, the file will be overwritten completely and replaced with a new file of the same name. This means that all data within the file will be lost, and you will reset the map to an empty space. For this reason, it is suggested that you use the largest terminal size possible.
Due to the way the exit system works, it is suggested you use numbered files starting with 0. For example, 'level0.txt, level1,txt, level2.txt, …' This will allow the exits to line up with the rooms they're named after, for example exit 3 will open 'level3.txt'. The levels will be loaded in order based on number, so it is also suggested that you don't have file names jump around in values. For example, if your directory contains files 'l0.txt', 'l1.txt', and 'l47.txt', l47 will generate map number 2, and will be accessible by exit 2. If an exit's number is larger than the number of files in the directory, then the player entering that exit will result in them completing the game. For

example, in the above example, the maximum file number is 2. For that reason, entering an exit numbered 3 or above will result in the player finishing the level pack. Creating exits is detailed in the later chart.
If you enter the name of a file which does not yet exist, you will be prompted to enter the width and height of the map. These dimensions must be entered as integer numbers. You will then be given a blank map to work with. If you enter a width or height that is too large to be displayed on-screen, the program will automatically adjust the dimensions to the largest possible size given the current size of your terminal. The program will then create a blank file, containing only space on which your character can walk.

Upon opening a file, you will be greeted with the contents of the map as they will appear in-game, as well as a green cursor. The cursor can be moved with the arrow keys, and will show you what appears on the map's 'init' array. That is, it will show you what appears in the actual txt file from which the map is loaded. The mapCreator allows you to place down objects wherever the cursor is, but only predefined objects. Here is a list of objects that can be placed, and what they do. The leftmost column shows the key (or keys) you can enter to place an object, the second is the name of the object, and the third explains the object:

1. Character	2. Name	3. Shown As	4. Function
-	Blank	Dark Blue Cell	Empty space, objects like enemies and the player can move into this space.
w	Wall	Black Cell	The player and enemies cannot walk into a cell containing a wall.
p	Player	Green Cell	This will create the player's starting position. If multiple are present, the last 'p' in the file will be the player.
k	Key	Yellow Cell	Certain rooms have a 'K' win condition, which requires a key to exit. The player must grab this key to complete these rooms.
Number (0 - 9)	Enemy	Red Cell with Number	This will create an enemy. Enemy 1 stays still and can be killed with a sword, Enemy 2 moves horizontally and can be killed with a bow, and enemy 3 moves vertically and can be killed with the shuriken. All other enemies stand still and will be invincible.
Place 'e' left of string of numbers	Exit	Cyan Cell with Number	By placing an 'e' to a cell immediately left of any string of numbers, which will appear as enemies, you can create an exit. Exits will contain the value of the string of numbers, and will attempt to move the player to that room if the player steps on them. For example, if a 4

			and 7 are placed right next to each other, then placing an e left of them will create exit 47, which will transport the player to room 47 when entered. If no room 47 exists, the player will instead complete the game.
s	Sword	Magenta Cell with Vertical Line	The sword is a weapon the player can pick up and use with the arrow keys. It attacks the cell directly next to the player in the direction they press, and kills type 1 enemies.
b	Bow	Magenta Cell with M	The bow is a weapon the player can pick up and use with the arrow keys. It shoots an arrow in the direction the player presses, and the arrow won't stop until it hits an object or the end of the map. Nothing but the arrow can move while one is firing. It can kill type 2 enemies.
h	Shuriken	Magenta Cell with +	The shuriken is a weapon the player can pick up and use with the arrow keys. Pressing any arrow key will attack all four of the spaces directly diagonal to the player. It can kill type 3 enemies.
Capital Letters	Printed Text	Black Cell with White Character	By entering uppercase letters (LIKE THIS), you can create printed text which will appear on the map and act as a wall. The cell will contain whatever character you placed there, so if you type an 'L' the cell will show an L.
q	Quit		Pressing Q will quit the program

Upon quitting the program, the player will be asked for one more piece of data: the win condition. Below is a chart showing each win condition, and how the player can beat them:

T	The player must simply reach an exit to progress.
K	In order to progress, the player must have at least one key. A key can be grabbed at any point in the game, and does not have to be present in all rooms with this win condition. As long as the player is able to grab a key in any room before or including one with this win condition, they will be able to progress.
S	The player must have a sword to progress.
B	The player must have a bow to progress.
H	The player must have a shuriken to progress.
E	The player must kill all enemies to progress.

Exits with smaller numbers than the current room can always be entered, while exits with larger numbers than the current room can only be entered if the win condition is met. For example, if room 2 has exits 1 and 3 and has win condition 'E', the player will be able to return to room 1 but will not be able to enter room 3 until all enemies have been destroyed.
After getting the win condition, the program will write all of your changes out to the file previously specified. It will create a text file containing the contents of the init array, and the win condition.

You may also directly edit maps by entering the level directories and manually changing data within the txt files, but doing so will allow you to make problematic maps which could cause issues, so it is suggested that you avoid this and only use the mapCreator. While you can still create 'impossible' levels using the included level editor (for example, you could create a room that doesn't even include a player), you won't be able to include ascii characters the game isn't programmed to handle, misalign the rows and columns in ways that create load time errors, remove the win condition, and so on. Using the mapCreator will ensure that the level you create won't break the program, even if it is impossible to actually win the level.