```
Introduction
o Project Overview
o Challenges and Solutions
o Search Algorithms
o Pseudocode/flowchart
o Test Evidence
o Conclusions
Use APA 7 format as the standard citation and reference style throughout the
assessment
```

Project Overview:

This project is a python game using a grid to move the player character around with the aim to find treasure while navigating through traps and obtaining power ups.

The treasure is created through classes using Object Oriented Programming allowing me to set an X and Y coordinate in the grid. To set these co-ordinated i called a function called Place_Treasure(). This function is used to decide the amount of treasure placed as well as where and what icon is displayed on the map. To place the treasure I used if statements to find out how big the user had set the grid to be then used a random number generator to place the treasure between 0 and the grid size. To decrease the possibility of the treasure being created next to the start point of 0,0 i ensured that either the X or Y coordinate randomly set wouldn't have either X or Y as a 0. This gives more space to the user to move around and avoid winning the game immediately.

The traps are also created using OOP for the same reasoning as the treasure as well as a very similar placement function using random numbers. The difference is shown when the user moves to a square where a trap is on. This will appear with a message asking the user to solve a riddle. If the user solves the riddle they are allowed to move on from the square. However if they fail the user will have their health decrease by 1. If the user's 5 health points reach 0 then the game ends and the user is told they have failed.

Power ups- The power ups, unlike the treasure and traps are displayed on the grid to incentivise the user to use the searching algorithms. The power-ups offer either a basic search of the user's Y axis from where they are telling the user where the closest trap or treasure is if this is applicable or the power-up will be for Binary Search where the user will be told what Y coordinate the treasure is on.

For the User another Class is created, similar to the others by setting an ID as well as X and Y coordinates but also having the player health and movement within the class as well. The health, as previously stated, is used as a consequence for falling into traps whereas the movement function within the class is used to ask the player where they would like to move to and to then change the users coordinates accordingly.

Challenges/Solutions

The biggest challenge I faced throughout this project was implementing the searching algorithms. With binary search, the challenge was that once the list was split which side should be searched next. This was an issue due to the unordered aspect of my original list. I sought to embed values within the list or set the entire grid to be one large list. The issue was that I was vastly over complicating the issue. This was solved very quickly once I realised I could just target the coordinates. The solution to this problem wasn't the code, it was changing my mind set about the problem. To try and combat this I decided to write down the issues with my current mindframe of coding and where precisely I was having trouble. I then stopped focusing on the path that I had taken in the code and started again with a different method in mind but the same end goal.Once I attacked the problem from a few different angles and wrote down the snags to each angle I had a more clearer picture of what I was doing wrong. To find the correct path, I compared the notes from each path and figured which paths complemented each other. For example if one method had errors with trying to use Math on a method or a string but another method used lists and numericals more then I would take out the problem code and instead include code from different tests. I have included some of the old code but commented out for the reader to see the different approaches I tried before finding a method that worked.

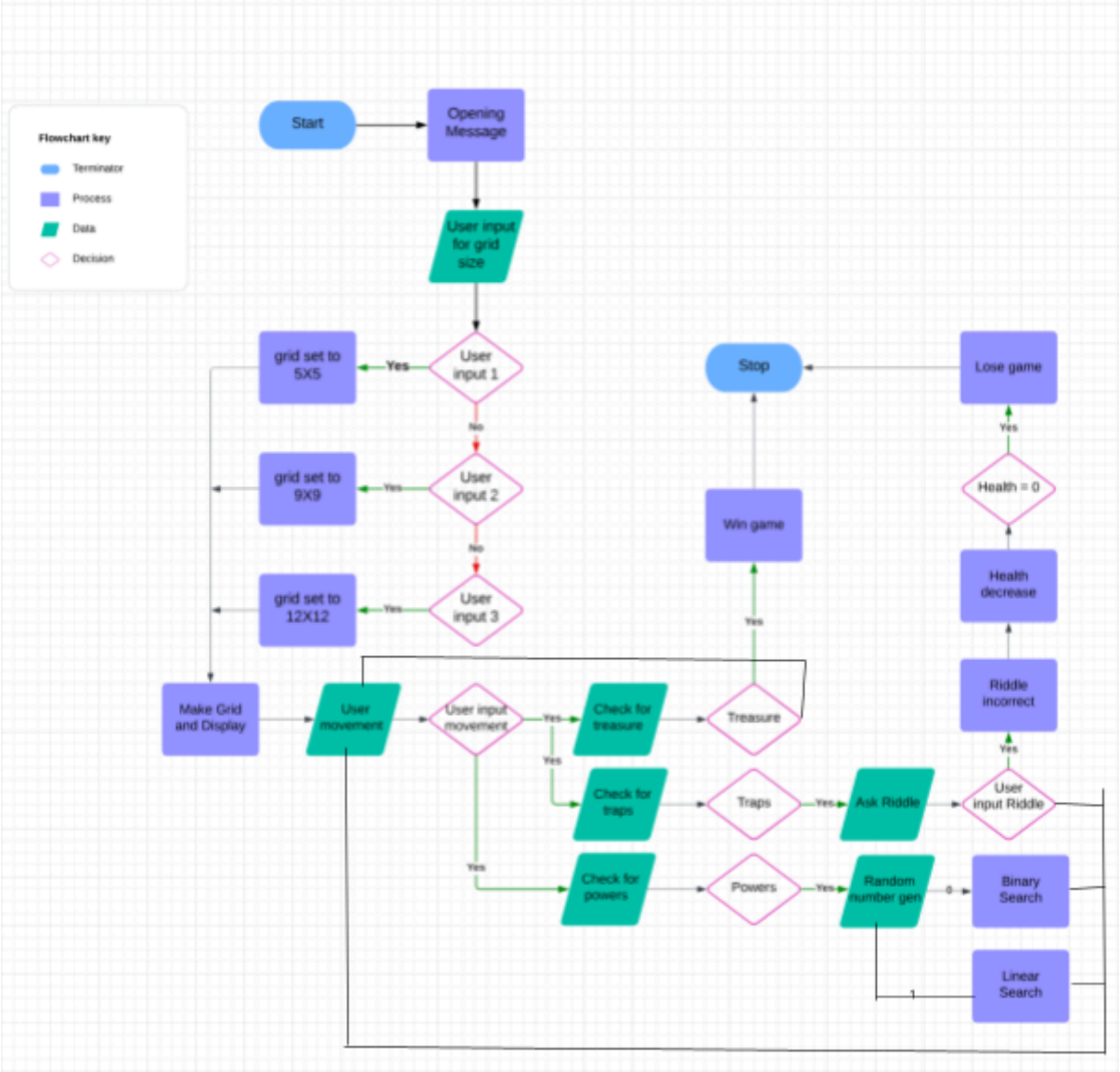
Searching Algorithms

In this project I have used two types of searches. A binary search and a default linear searching algorithm.

My linear searching algorithm uses the comparison that my binary search does to check if that number is the one the code is looking for but without the need of splitting the list. Due to the small scale of values binary search is not needed, instead the linear search just travels through the list checking each number. This is less efficient however for the scale of the project this is perfectly acceptable and works as needed.

As it currently stands the binary search will create a list from 0 to the grid size. From here the list is split and the middle number is checked against the treasure. If it is not the middle number then the list is reorganised by changing the mid point to target the value of the chosen coordinate. Depending on if the value is higher or lower than the midpoint, changes which side of the midpoint is then searched. This happens again and again until the midpoint is the target the algorithms are looking for. I decided to have my list only display one axis of the treasure so the player doesn't find it straight away.

FlowChart


Flowchart key
Terminator
Process
Data
Decision
Start
Opening Message
User input for grid size
User input 1
Yes
grid set to 5X5
No
User input 2
Yes
grid set to 9X9
No
User input 3
Yes
grid set to 12X12
Make Grid and Display
User movement
User input movement
Yes
Check for treasure
Treasure
Yes
Win game
Stop
Check for traps
Traps
Yes
Ask Riddle
User input Riddle
Yes
Riddle incorrect
Yes
Health decrease
Health = 0
Yes
Lose game
Yes
Check for powers
Powers
Yes
Random number gen
0
Binary Search
1
Linear Search

Test Evidence

In testing different individual components before integrating them I use separate py files. Within the code i have left the commented out previous code that i used to show how i tested then edited the code to fit the working code you use now. There are no issues with the current code as seen below.

Test 2.py > ...

```python
mrange = 5
bs_x = 1
bs_search_list =[]
x = 0

get_t_y = 3


# for i in range(mrange + 1):
#     bs_search_list.append(i)
#     i += 1
#     bs_y = bs_search_list[x]
#     print(bs_y)
#     x += 1
# print(bs_search_list)
# bs_search_list.sort()
# print(bs_search_list)
for i in range(mrange):
                    bs_search_list.append(i) # making the list that will be used for BS search
                    i += 1
                    low = 0
                    high = len(bs_search_list)
                    v = 0
                    # bs_y = bs_search_list[x]
                    # print(bs_y)
                    # x += 1

                    # if v == 0:
while high >= low:

                mid = (high + low) // 2

                if bs_search_list[mid] == get_t_y:
                    print (f"Treasure is on y: {get_t_y}")
                    break

                elif bs_search_list[mid] > get_t_y:
                    high = mid - 1

                else:
                    low = mid + 1

                # if bs_x == power.get_p_x() and target == power.get_p_y():
                #     print (f"Power found at {bs_x}, {i}.")
                #     v =+ 1

                # elif bs_x == trap.get_tr_x() and target == trap.get_tr_y():
                #     print (f"Trap found at {bs_x}, {i}.")
                #     v =+ 1


                # else:
                #     i += 1
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  COMMENTS

However, you will come across dangerous traps and puzzles.
Will you survive or will the island take you?


How big would you like the map to be?
1. Small
2. Medium
3. Large
```

```python
# def binary_seach(bs_list, low, high, T):
#     global mid
#     bs_list = map_data
#     if map_size_q == "1":
#         high = str(high)
#         high = 5
#         low = str(low)
#         low = 0
#         mid = str(mid)
#         if high >= low:
#             mid = (high + low) // 2

#             if bs_list[mid] == T:
#                 return mid

#             elif map_data[mid] > T:
#                 return binary_seach(bs_list, low, mid -1, T)

#             else:
#                 return binary_seach(bs_list, mid + 1, high, T)
#         else:
#             return -1
#     BS_result = binary_seach(bs_list, 0, len(bs_list)-1, T)

# def binary_search(bs_x, y):
#     for user in users:
#         bs_x = user.get_x()
#         for i in range(mrange):
#             if bs_x == power.get_p_x() and i == power.get_p_y():
#                 print (f"Power found at {bs_x}, {i}.")

#             elif bs_x == trap.get_tr_x() and y == trap.get_tr_y():
#                 print (f"Trap found at {bs_x}, {i}.")

#             else:
#                 i += 1
```

```
Trap placed
Powers placed
User starting at 0, 0.
[ U ][ 0 ][ 0 ][ 0 ][ P ]
[ 0 ][ 0 ][ P ][ P ][ P ]
[ 0 ][ 0 ][ 0 ][ 0 ][ 0 ]
[ 0 ][ 0 ][ 0 ][ 0 ][ 0 ]
[ 0 ][ 0 ][ 0 ][ P ][ 0 ]
Enter user movement. UP DOWN LEFT RIGHT:
```

```
Map initialized
[ 0 ][ 0 ][ U ][ 0 ][ P ]
[ 0 ][ 0 ][ P ][ P ][ P ]
[ 0 ][ 0 ][ 0 ][ 0 ][ 0 ]
[ 0 ][ 0 ][ 0 ][ 0 ][ 0 ]
[ 0 ][ 0 ][ 0 ][ P ][ 0 ]
You have found a trap!
 Answer a riddle or lose health.
Riddle: What has a face and two hands but no arms or legs?
```

Conclusion

In conclusion, I have found this project to be a test of my skills in coding to find new methods to fit the same problem. The challenges I have faced are mainly in integrating the searching algorithms to meet the criteria set in the brief. However I've enjoyed the creative freedom that has come with the game creation and being able to include some of my own ideas such as the riddle system in the traps. I would state that I have developed a functional grid and treasure finding game. Meeting the MVP and coding optional improvements.

References

MyBib Contributors. (2019, May 26). APA Citation Generator – FREE & Fast – (6th Edition, 2019). MyBib. https://www.mybib.com/tools/apa-citation-generator
- This was used to format references in the preferred University Style.

Microsoft Copilot. (2024). Microsoft Copilot. Copilot.microsoft.com; Microsoft. https://copilot.microsoft.com/

- This was used to summarise the brief as well as pinpoint errors when the python debugger did not display errors despite the program not running as planned.

W3Schools. (2019). Python Classes. W3schools.com. https://www.w3schools.com/python/python_classes.asp

- This was used to write the classes used and integrate it into my code

W3schools. (2019). Python Lists. W3schools.com. https://www.w3schools.com/python/python_lists.asp

- This was used for the grid, searches and most functions as well as the framework to have multiple treasures, traps, powers and users.