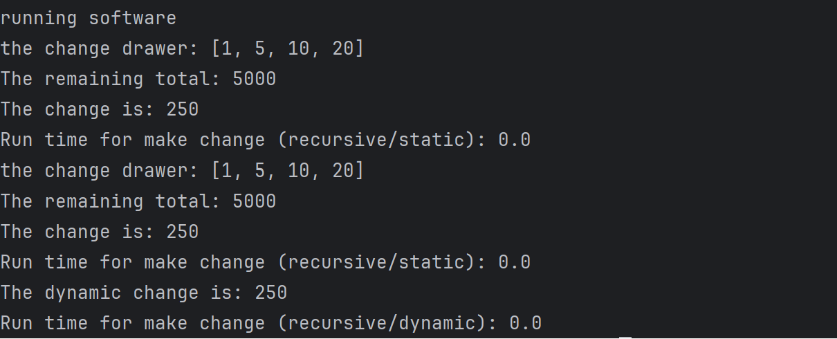
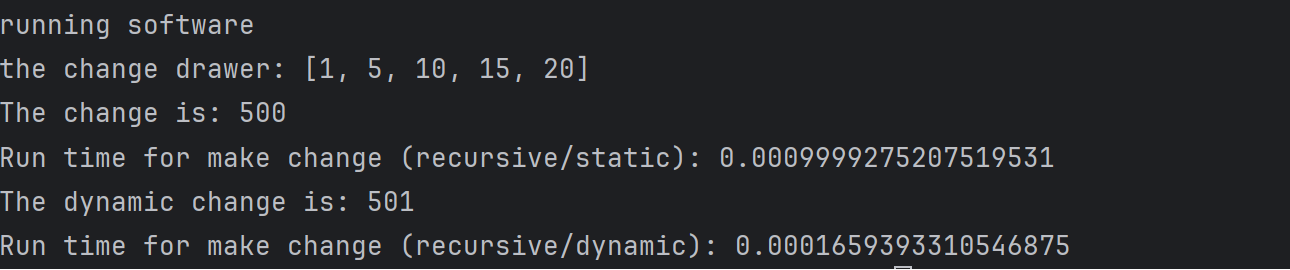
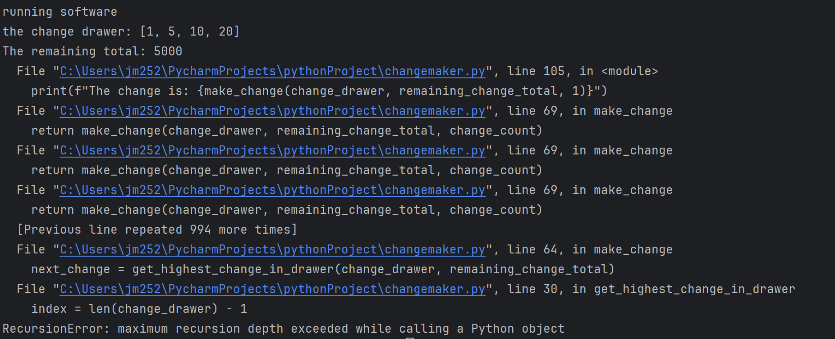
For this assignment I wrote two functions, one that used recursive programming to determine the minimum number of unique coins required in the cache drawer. For the second function, I made a copy of the first, with the change that it uses a cache to avoid recusion whenever possible. For this I used a while loop, so that the cache hits could be chained together without calling the recursive function. This seemed the only way I could get it to work, where it wasn’t going into the recursive function call but still processing forward. Its tricky to avoid the call, because if you don’t call it, then the whole thing stops working.

The cache works by loading and saving values from a dictionary to a json in a file. Below you can see the increased speed of the cache, when using the previous values it sped up the runtime of the software’s recursive calls. The cache seemed to generally speed things up. Although, it was sometimes hard to tell, for the reasons I’ll explain below.

Unfortunately, for runs below 5000 size would not give me a python run time of above 0.0 (for some reason for these it just shows 0.0 (screenshot below). This seemed to be an intermittent issue.

For runs above 10000, the normal recursion generated an error (shown below for as low as 5000 change size). The dynamic software seemed able to climb to higher numbers tho, probably because cache hits allow it to avoid recursion.

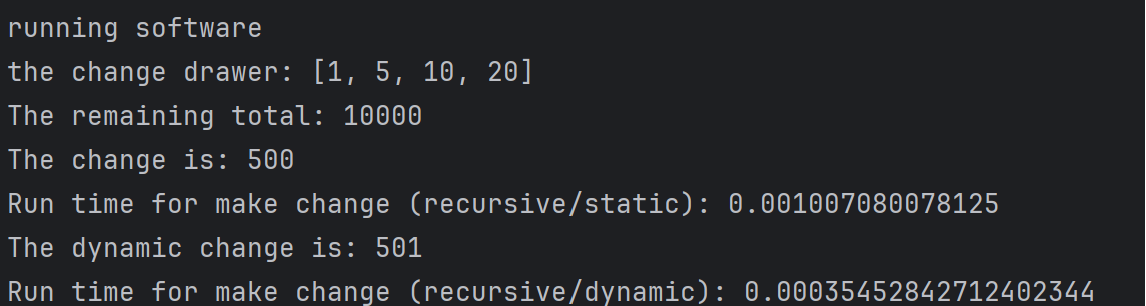


Here you can see what happends when the normal recursive software is asked to go > around 5000 change size, it errors out with too much recusion.

The dynamic software for example (shown below) was here able to hit a run of 10000, twice the size of the threshold where the above software was crashing. This is a big advantage produced using the Dynamic Software method.

A black screen with white text

Description automatically generated

The Dynamic Software outperformed the other software generally; however, sometimes it was tough to tell because of the 0.0 run time for both occurring intermittently. For this reason, I did not feel my data was reliable enough to generate a graph, but I have shown some screenshots below, where the dynamic software clearly outperformed the static software.

A computer screen with white text

Description automatically generated