Algorithms for Data Science
Assignment 3 Portfolio Optimization

In this assignment, you will be using dynamic programming to select a set of investment options that maximize your return on investment. You will be given a file of investment options along with the estimated return on investment for each one. You also have a given amount of money to invest. The specific file of investment options you will be given will vary, but you will always have the following three pieces of information you can obtain from the file:

```
InvestmentName, InvestmentCost, EstimatedReturnOnInvestment
```

For example:

```
A,10,60
B,20,100
C,30,120
```

The input file will be state_zhvi_summary_allhomes.csv. The first row contains the headers and the second row contains the average for the United States. Do not include the United States row, start the import on the third row (California). There is a "region name" column that is the state and it will be used as the investment name. There is the "zhvi" column and that is the average price of homes in that state. This will be your investment cost. Finally, "10year" is the average return on homes in that state and that column is a ratio (not percentage, take times 100 to get percentage). Multiply that 10year column by the zhvi column to get your estimated return on investment in dollars. For example, California will be:

InvestmentName, InvestmentCost, EstimatedReturnOnInvestment
California, 547700, 23625

Start by putting all your code in a file called portfolio.py. Create a function called loadInvestments that takes in the investmentFilename and returns a list of possible investment options: name, cost, and estimated return. You will be given a file of actual investment options along with specific directions for that file on how to obtain the three pieces of information necessary for this assignment.

Then make a function called optimizeInvestments that takes the list of possible investments along with the amount of money available to spend. This function should return both the optimal return on investment amount as well as the actual investments selected to achieve this optimal return. Implement this function using dynamic programming.

For dynamic programming, one needs a recursive breakdown of the problem. We have a number of possible investments and an amount we are able to spend. At each step we can decide to either include investment *x* or exclude it. If we include it, then we are left with a smaller problem of finding the optimal return with all the possible investments without investment *x* with a smaller amount we are able to spend (because we purchased investment *x*). If we exclude it, then we are left with a smaller problem of finding the optimal return with all the possible investments without investment *x* but with the same amount of money to spend as we had before.

A DAC solution could be attempted, but it ends up having too much repeated work. Therefore, we will implement our solution with dynamic programming. Note that this is a 2-D problem (number of investments and amount of money to spend). Thus, we will need a 2-D table. The second step is filling in the base cases. Note that when we don't select any investments, no matter how much money we have to spend (top row), then we get no return on investment. This is a natural base case. The third step is to identify the goal location in the table. This will be the position where we are allowed to include all investments with our full allocation of funds to spend. This will be the lower-right corner. The last step in computing any optimal table is to determine the order in which to fill in the table, keeping in mind that one always needs all the results for the recursive subproblems available to fill in the next location.

After the optimal table is computed, a traceback table needs to be added to the implementation so we can find the actual investments that produce the optimal. Recall that traceback tables store the "winning" options at each step and are then used at the end to trace back through these winning options. Your code should return the optimal return on investment number as well as a list of the investment names used to obtain this optimal number.

Note that this is a similar problem to the 0/1 Knapsack problem, and you should use that idea to help guide you through your implementation.

I included a file called zhvi-short.csv for testing. It runs much faster than the full 50 states. If you have a maximum of $15 to invest and your run portfolio.py against zhvi-short.csv, you should pick Illinois and California to maximize your return. You can easily calculate the answer by hand to confirm.