Coding Task 3
Theme: Dynamic Programming

Rules:

- Use **Python 3.7** or higher in **Google Colab**. It comes with pre-installed packages like numpy, time, etc.
- Submission should be a single **.zip** file (no other extensions like .rar, .gz) Keep this in mind.
- Submission should contain
  1. Code files in **.ipynb** format. Use separate **.ipynb** files for separate questions. **DO NOT** use .py files. We will test your code in Google Colab.

  2. **Also, submit your solutions in the format ROLLNO_Qx.py for each question. Change Q suffix according to question number. This will be used for checking plagiarism.**

  3. README file describing how to run the code. TAs will run your code to see if they are really working.

  4. A write-up in **.pdf** format. It should contain answers to the questions with appropriate screenshots or test results from the output of your code. Treat this as a mini-report that TAs will evaluate after your code successfully runs.
- **DO NOT** use built-in functions for cases where you have been asked to implement that exact same functionality. You can use built-in functions if you want to test whether your designed function produces the correct output.

Q1. **Fractional Knapsack Problem**

In tutorial, we discussed the 0-1 Knapsack Problem. In 0-1 Knapsack Problem while picking up items to put into the knapsack, we are not allowed to pick up parts of an item. We have to either pick it or discard it.

In the case of Fractional Knapsack Problem, you are allowed to pick up certain portions of the item instead of the whole item if the need arises. Similarly, the value of the partially picked item will depend upon what fraction of the item has been picked up.

Take a fixed constant factor which will denote the fraction of an item you can pick up. Suppose if you take the factor as 0.3, then either you can pick up the whole item or 0.3 parts of the item

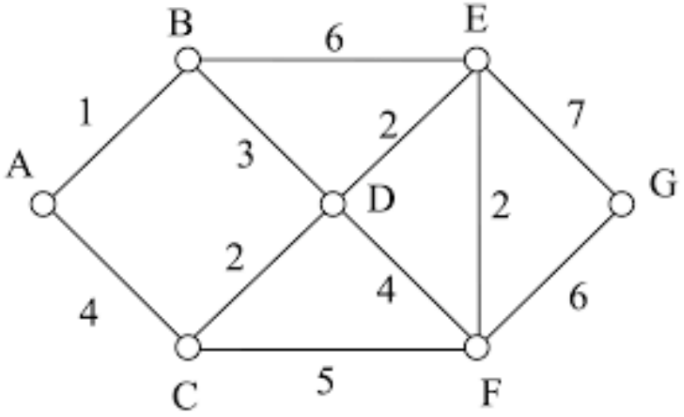
(the value also becomes 0.3 of the entire value of the item) or you don't pick it up at all. You are free to choose that factor for your implementation.

Implement the Fractional Knapsack Problem using:

a. Naive recursive approach (Top Down without memoization)
b. A Dynamic Programming Approach (Bottom Up)

Using proper example inputs show that the algorithm picks up fractional parts of an item when required.

Q2. **Travelling Salesman Problem**



Take a look at the graph above.

Suppose you live at A. All the other vertices are homes of your friends. You are leaving the country tomorrow and have a lot of pending official work. But you plan to visit every one of your friend's homes (they can't come for reasons not important in the problem setting) before flying out of the country. So you plan to visit each of your friends exactly once and complete your entire journey as fast as possible.

Implement an algorithm to do so and as output print the path you choose ultimately. This is a version of the TSP algorithm you were taught in the theory lecture.

NOTE: You may use Python's inbuilt data structure or implementations from your previous assignment solutions for traversing graphs. But the main task should be your own implementation.