Data Structures Lab 4 Name:_____

Objective: Practice designing program with multiple functions.

Lab: Your team (2 or 3 students) are to **design** (you do not need to implement/code it) a Python program that allows a user to play a video poker game that uses dice. The base set of rules is as follows:

- A player starts with $100.
- Each round costs $10 to play. This amount is subtracted from the user's money at the start of the round.
- The player starts the round with five randomly rolled dice.
- The player gets two chances to enhance the hand by rerolling some or all of the five dice.
- At the end of the hand, the player's money is updated according to the following payout schedule:

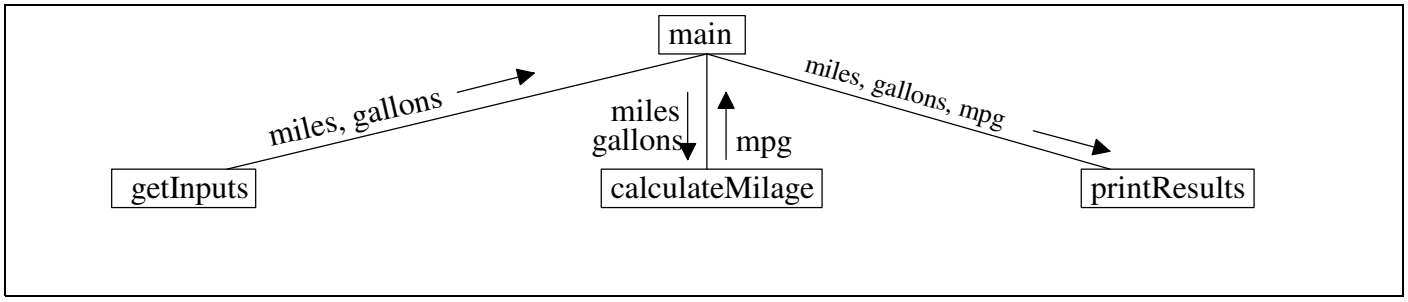
Hand	Example	Payout
Two Pairs	4, 4, 3, 6, 6	$ 5
Three of a Kind	2, 4, 1, 1, 1	$ 8
Full House (a pair and a three of a kind)	3, 5, 5, 5, 3	$ 12
Four of a Kind	3, 3, 3, 3, 6	$ 15
Straight (1-5 or 2-6)	5, 2, 3, 1, 4	$ 20
Five of a Kind	3, 3, 3, 3, 3	$ 30

A text-based user-interface might go something like this:

```
Welcome to video poker!
You currently have $100.
It costs $10 to play. Do you wish to try your luck (y or n)?  y
Roll 1 Dice: [6, 4, 4, 2, 4]
Positions:    1  2  3  4  5
Enter the positions of the dice you want to reroll (or Enter to stop):  1  4
Roll 2 Dice: [3, 4, 4, 4, 4]
Positions:    1  2  3  4  5
Enter the positions of the dice you want to reroll (or Enter to stop):  1
Roll 3 Dice: [2, 4, 4, 4, 4]
Four of a Kind.  You win $15!
You currently have $105.
Do you wish to try your luck (y or n)? y
. . .
```

For these relatively small program, I recommend the following top-design methodology:

1) Think about where the program would need to loop and what task is being performed by each loop
2) Think about how these programs might be nested in a "single main program," but don't actually write it
3) Split the "single main program" into functions such that:
    - a main function acts as an outline of the program by calling other functions
    - each function performs a single, well-defined task
    - each function contains at most one loop (the body of the loop might call other functions containing a loop)
4) Draw a *structure chart* (p. 207 in text) showing the calling structure between the functions with information (i.e., parameters) flow shown. Below is a sample structure chart for a simple miles-per-gallon program



Each group should hand in a single structure chart and brief (sentence or two) description for each function. (Make sure every group members' name is on the structure chart)