Homework Program 3

In this assignment you will write a Matlab script file that simulates a series of simple betting games. Each game can be interpreted as a race with $N > 1$ contestants. The player is given a budget, and will place a bet (representing a dollar amount) on each contestant to win the race. The computer generates the (random) racing times, determines the winner and the payout to the player corresponding to the bets placed on each contestant.

The program consists of the following main steps.

1. The computer generates average, or mean times for each of the N contestants. That is, the actual racing times are random, but some contestants are faster than others, on average. This means the random racing time for a particular contestant will be biased higher or lower according to its average time.
2. The computer determines and presents the odds, which determines the payout for a bet placed on each contestant, assuming that contestant wins the race. That is, the payout for a contestant is given by the odds for that contestant times the corresponding bet on that contestant to win, where the odds (multiplier) is determined by the average race times (a simple formula). If the contestant is relatively fast (low average race time) the odds multiplier will be relatively small, and the slower the contestant (on average) the larger the odds and payout will be for betting on that contestant.
3. The player enters bets on each contestant to win. The sum of the bets cannot exceed the current budget.
4. The computer generates the race times for each contestant by drawing random numbers and determines the winner and payout. The random numbers (race times) will be biased higher or lower based on the average values specified in the first step.
5. The player's budget is updated after the race by adding any payout, if a bet was placed on the winner, and subtracting the bets.

The player is assigned an initial budget, and can allocate all or part of the budget as bets across the contestants. The player can continue to bet on successive races (that is, continue with another random trial) until either the budget drops below a minimum amount required to place a bet, or grows beyond a given cap.

Begin your Matlab script with your version of the comment:

```
% EA 1, Homework program assignment 3
%
% Name:         Nakamoto, Satoshi
% Date:         10/12/2023
```

Write Matlab statements corresponding to the following steps:

1. Set the number of contestants `num_racers = 4` and prompt the player to enter the budget variable `budget`. Check to make sure `budget >= 1`, which is the minimum bet for the assignment. If `budget < 1`, then print a message to inform the player, and continue to prompt for the budget. Also for this assignment, you can fix the cap on total winnings to be `budget_cap = 2*budget`.
2. Define an array `mean_times` containing the mean times for the contestants by drawing a random number between 1 and 3. Use the function `rand` to do this. Note that `rand` generates a random number between 0 and 1. (The value is equally likely to appear anywhere within the interval $(0, 1)$.) Hence for constants `a` and `b`, `a*rand + b` shifts the interval from $(0, 1)$ to (`b`, `a+b`).

3. Determine the payouts for each contestant as follows: if contestant `n` wins, the payout is `odds(n)*bets(n)` where `bets(n)` is the bet placed on contestant `n` and `odds(n) = 1/prob_win(n)`, where `prob_win(n)` is the chance (probability) that contestant `n` wins. (Hence the payout increases as the chance of winning decreases.) For the way the race times will be determined,
   `prob_win(n) = inv_time(n)/sum(inv_time)`
   where `inv_time(n) = 1/mean_times(n)`. That is, the probability that contestant n wins is $(1/T_n)/\sum_{i=1}^{N}(1/T_i)$, where T_i is the average race time for contestant i. So the smaller T_i is, the higher the probability of winning.

   For example, if there are $N = 2$ contestants and the mean times are $T_1 = 1$ and $T_2 = 1.5$, then the chance that contestant one wins is 0.6 and the payout for winning is 5/3 times the amount bid. For contestant two, the chance of winning is 0.4 and the payout is 2.5 times the amount bid.

4. Print the array `odds` for the contestants and prompt the user to enter bets. The bets should be an array with N elements. Check that the number of entries matches the number of contestants, that the sum is less than the current budget, and that all bets are nonnegative. If the bet array is invalid, then prompt the user to bet again.

5. Generate the random time for contestant `n` with the command
   `times(n) = -mean_times(n)*log(rand);`
   except replace this with the vector version that assigns the entire array `times`. The function `-log(rand)` generates a random positive real number that can be arbitrarily large (if `rand` is near zero), although the probability of generating a value near some $x > 0$ decays exponentially as x increases. Multiplying by `mean_times` biases the outcomes up or down according to the mean times for the contestants. Print the times and the winner.

6. Update the budget by adding the payout (`odds(n)*bets(n)`) if a bet was placed on the winner, and subtracting the total amount bid.

7. Check the budget against the minimum bet and the cap to see if the player is able to continue to bet on a new race. If not, print a correponding message and terminate the game. Otherwise, prompt the player to continue before starting the next race. That is, the player has the option of terminating the game.

8. Upon exiting print a summary showing the bet and updated budget after each race, as shown in the examples that follow.

Run your program as the player with four contestants, a budget of 10, and a budget cap of 20. Copy and paste outputs corresponding to three outcomes:

1. Your budget drops below one, the minimum bet.

2. Your budget exceeds the budget cap.

3. You terminate the program by choosing not to continue when prompted. In this part, enter the first vector of bets incorrectly.

Note that the way the odds are determined, the average return on any bet is zero. That is, *on average*, the budget will not change after each bet no matter how the bets are distributed across contestants. However, there will be substantial variations in the budget from race to race due to randomness that depend on how the bets are made. In particular, the variation will increase with the size of the bets, and by allocating a larger fraction of bets to particular contestants.

Example program runs are shown below. As in previous homeworks, submit your program as a script (.m) file and show your outputs as comments appended to the script file.

Example Program Outputs:

```
>> race
Enter initial budget (>1): 10

Contestant odds:
6.25  5.63  2.96  3.08
Input bets:
[5 0 5 0]
Times:
0.64  1.52  1.09  0.68
The winner is contestant number 1.
Your budget is now $31.27
Congratulations, you exceeded your cap!

Race 1: bet $10.00, budget $31.27


>> race
Enter initial budget (>1): 10

Contestant odds:
3.53  3.37  5.80  4.04
Input bets:
[3 3 0 0]
Times:
4.30  3.12  0.33  1.39
The winner is contestant number 3.
Your budget is now $4.00
Continue? (1=y, 0=n) 1

Contestant odds:
4.58  2.37  5.10  6.13
Input bets:
[2 2 0 0 0]
Bets are invalid. Try again...
Input bets:
[2 2 0 0]
Times:
0.37  0.14  1.25  0.55
The winner is contestant number 2.
Your budget is now $4.74
Continue? (1=y, 0=n) 1

Contestant odds:
3.67  7.48  2.88  4.05
Input bets:
[2 2 0 0]
Times:
1.48  0.29  0.83  1.28
The winner is contestant number 2.
Your budget is now $15.69
Continue? (1=y, 0=n) 1
```

```
Contestant odds:
5.88  3.20  3.78  3.94
Input bets:
[13 0 0 0]
Times:
0.60  3.15  0.25  2.52
The winner is contestant number 3.
Your budget is now $2.69
Continue? (1=y, 0=n) 1

Contestant odds:
3.81  3.29  5.54  3.94
Input bets:
[0 0 2 0]
Times:
0.72  4.73  7.21  0.25
The winner is contestant number 4.
Your budget is now $0.69
Your budget is less than the minimum bet.

Race 1: bet $6.00, budget $4.00

Race 2: bet $4.00, budget $4.74

Race 3: bet $4.00, budget $15.69

Race 4: bet $13.00, budget $2.69

Race 5: bet $2.00, budget $0.69
>>
```