Ryan Mysliwiec
HW 10 – 2

<u>Algorithm</u>

```
MinCost(price, c)
     weight = [0] in range(price)
     fridgeCapacity = 0
     i = 0
     min = -1

     while i <len(price)
          if fridgeCapacity == 0
               weight[i] = 50
          k = i + floor((c - fridgeCapacity)/50) - 1
          j = i

          while j <= k
               if price[i] < price[j]
                    counter++

          weight[i] += counter*price[i]
          fridgeCapacity += (weight[i] - 50)
```

<u>Description</u>

The algorithm will search k* days after the first day for the cheapest meat price in that range. When the day with the cheapest price is found, the algorithm will decide that is the day to purchase enough meat to fill the fridge. The day after is then chosen as the new starting day for the same procedure. The algorithm will continue to do the same process as above until it reaches the last day. At this point, if there is any food in the fridge, it will be at least 50 lbs. Otherwise, if there is no food, 50 lbs. will need to be purchased.

*k is the number of possible days that the fridge could have enough food to feed the lions when the first day of that group has the fridge fully-stocked.

<u>Proof of Correctness</u>

G = {g1, g2, …, gk} is the greedy solution: the cheapest purchase of meat possible
O = {o1, o2, …, ol} is an optimal solution; k < l

Assume i is the first element such that g_i =/= o_i
Therefore, either

1) $g_i < o_i \rightarrow$ o_i will conflict with $o_{i-1}=g_{i-1}$
2) $g_i > o_i \rightarrow g_{i+1} <= o_{i+1} \rightarrow \ldots \rightarrow g_k <= o_k \rightarrow o_{k+1}$
    a. It is not possible for o_{k+1} to be non-conflicting with o_k

Because of the above statements, it is clear that there is no solution more optimal than the greedy algorithm.

<u>Runtime</u>

The algorithm will iterate through the days at most n times. This will happen when c = 50, so that you would need to purchase food every day, regardless of price, since there is no food in the fridge. This also shows that the algorithm is dependent on the size of the fridge, c, as that will determine how many groups of days the algorithm will need to iterate through. Therefore, the algorithm runs in O(cn).