Written Assignment 4
Greedy Algorithms.

1. Input : a] Array stops [contains the source, destination and all the gas stations.
b] n = length of array stops
c] m = miles that the car can travel when tank is full.

Output : minimum number of stops

Algorithm :-

```
procedure minstops (stops, n, m):
1       totalRefills = 0
2       CurrentRefill = 0
3       while currentRefill < n-1
4           lastRefill = currentRefill
5           while CurrentRefill < n-1 and
6               stops[currentRefill +1] - stops[lastRefill] ≤ m
7                   CurrentRefill += 1
8           If CurrentRefill < n-1
9               totalRefill += 1
10      return totalRefill.
```

Correctness & Optimality

- The algorithm is correct & optimal because we always refill at the farthest reachable gas station considering we always fill the full gas tank at every stop.
- Since we will always go till the farthest gas station we will make the minimum number of stops.

2. Input : n events
g profit associated with each event
t 'scheduled by' time for each even [deadline].

Output : Optimal Schedule that maximizes profit

Algorithm:

procedure schedule (n)

1. Sort jobs according to profit
i.e., $g1 \geq g2 \geq \ldots \geq gn$
2. for i in range (1, n):
3. Schedule job i in the latest possible free time slot [by its deadline]
4. If there is no slot for i then & continue

Proof :-
Since the most profitable jobs are selected first and scheduled before their deadline the ~~algorithm~~ algorithm is correct and optimal as ~~the~~ it maximizes the profit.

3. Input :- Set $x_1 \leq x_2 \leq \ldots \leq x_n$ points on the real line.
Output :- list of unit length intervals covering all points.

Algorithm :
1. Let x be the smallest element in the set which is not in any list.
2. while $x_i <= x+1$:
3. add x_i to the list of x
4. If set is exhausted [all elements have been considered]
5. Exit
6. Else GOTO step 1

Proof :-
Since the set is already sorted we select the smallest element say 'x'. Then we create a list ~~with~~ in with interval $[x, x+1]$ and add ~~numbers~~ the points belonging to this

interval. Once we get to a number which is greater than $x+1$, we make that number/point the new x and create a new list s with interval $[x, x+1]$ and repeat the above procedure till all the numbers/points in the set are covered.

Therefore the algorithm is optimal & correct as it yeilds the smallest set of unit length closed intervals that cover all points.

4 Input : n [change required]
Output : least number of coins.

Algorithm :

```
procedure minCoins(n):
1.      change = [25, 10, 5, 1]
2.      list_of_coins = [ ]
3       for i in change:
4.          while n >= i :
5               n = n - i
6.              list_of_coins.append(i)
7.      returns list_of_coins
```

Prog :-

Since the array change is sorted in descending order the coins with the highest value are added to the list first. Therefore we output the least number of quarters, dimes, nickels and pennies.