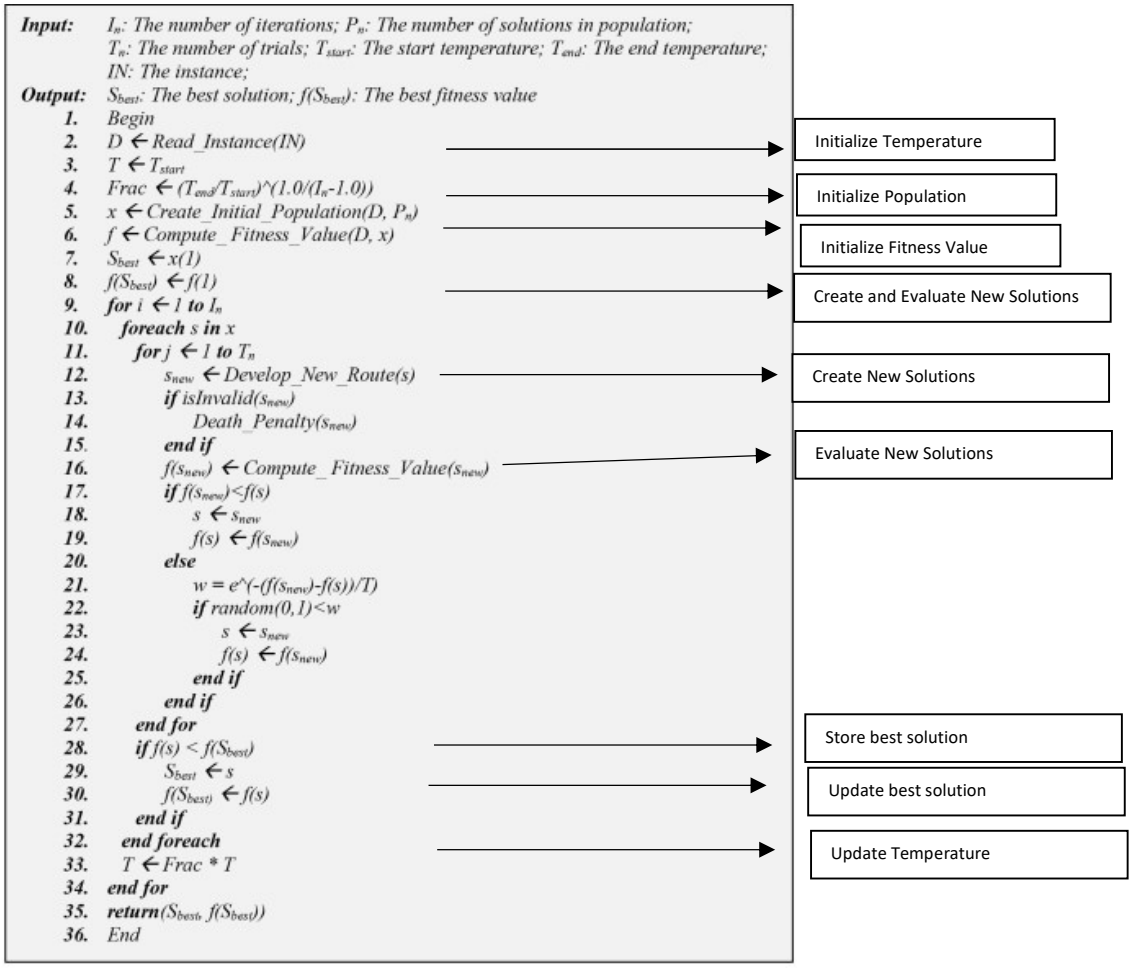
Question 2:

1) Source: A population based simulated annealing algorithm for capacitated vehicle routing problem, İLHAN İLHAN
   The writer's purpose is solving a Capacitated Vehicle Routing Problem using metaheuristic algorithms. Vehicle Routing Problem is identifying the best route to reduce distribution costs and improve the quality of service provided to customers. The writer is using randomly insertion and reversion operators on 23 instances on MATLAB for testing and analyzing with other algorithm.



2) Indicating steps of code is commanded by the writer. I just looked and understand the code.

The code source: https://yarpiz.com/372/ypap108-vehicle-routing-problem

In this source, you can find a lot of examples. They are using matlab.

```
mu = 0.5;      % Mutation Rate

sigma = 0.1*(VarMax-VarMin);    % Mutation Range (Standard
Deviation)

%% Initialization

% Create Empty Structure for Individuals
empty_individual.Position = [];
empty_individual.Cost = [];

% Create Population Array
pop = repmat(empty_individual, nPop, 1);

% Initialize Best Solution
BestSol.Cost = inf;

% Initialize Population
for i = 1:nPop

  % Initialize Position
  pop(i).Position = unifrnd(VarMin, VarMax, VarSize);

  % Evaluation
  pop(i).Cost = CostFunction(pop(i).Position);

  % Update Best Solution
  if pop(i).Cost <= BestSol.Cost
    BestSol = pop(i);
  end

end

% Array to Hold Best Cost Values
BestCost = zeros(MaxIt, 1);

% Intialize Temp.
T = T0;

%% SA Main Loop

for it = 1:MaxIt

  for subit = 1:MaxSubIt

    % Create and Evaluate New Solutions
    newpop = repmat(empty_individual, nPop, nMove);
    for i = 1:nPop
      for j = 1:nMove

        % Create Neighbor
        newpop(i, j).Position = Mutate(pop(i).Position, mu,
sigma, VarMin, VarMax);

        % Evaluation
        newpop(i, j).Cost = CostFunction(newpop(i, j).Position);

      end
    end
    newpop = newpop(:);

    % Sort Neighbors
    [~, SortOrder] = sort([newpop.Cost]);
    newpop = newpop(SortOrder);

    for i = 1:nPop

      if newpop(i).Cost <= pop(i).Cost
        pop(i) = newpop(i);

      else
        DELTA = (newpop(i).Cost-pop(i).Cost)/pop(i).Cost;
        P = exp(-DELTA/T);
        if rand <= P
          pop(i) = newpop(i);
        end
      end

      % Update Best Solution Ever Found
      if pop(i).Cost <= BestSol.Cost
        BestSol = pop(i);
      end

    end

  end

  % Store Best Cost Ever Found
  BestCost(it) = BestSol.Cost;

  % Display Iteration Information
  disp(['Iteration ' num2str(it) ': Best Cost = '
num2str(BestCost(it))]);

  % Update Temp.
  T = alpha*T;

  sigma = 0.98*sigma;

end

%% Results

figure;
%plot(BestCost, 'LineWidth', 2);
semilogy(BestCost, 'LineWidth', 2);
xlabel('Iteration');
ylabel('Best Cost');
grid on;
```