Part 1:

• Simulated Annealing with T=0 at all times and omitting the termination test

```
function SimulatedAnnealing(problem)
        current=initial state of problem
        t=1
        loop
                T = 0
                next = random successor of current
                δE = next.VALUE - current.VALUE
                Because T=0 so the next value is worse than the current value.
                Update current=next with some prob (e δE T )
                t=t+1
```

• Genetic algorithm with population of size 1

```
// start with an initial time
t = 0;
// init a usually random population
init population P (t);
// evaluate fitness of all initial individuals
evaluate P (t); two selected parents will be the same individual
// test for termination criterion (time, fitness, etc.)
while not done do
        // increase the time counter
        t = t + 1;
        // based on fitness select a sub-population for mating
        P' = select parents P;
        // recombine the "genes" of selected parents
        recombine P';
        // perturb the mated population stochastically
        mutate P';
        // evaluate new fitness
        evaluate P';
        // select the survivors from actual fitness
        P = P';
return random
```

• Modify the Hill Climbing algorithm so that it implements random restarts H.C

```
current = problem.INITIAL_STATE
loop
        function hillClimbing(problem)
        current = problem.INITIAL_STATE
        loop
                neighbor = highest-value successor of current
                if (neighbor.VALUE<=current.VALUE) then
                        return current.STATE
                current = neighbor
        current = neighbor
```

Part 2: