Oleg Perchyk
CS355 – Beyond Search

Beyond Search Homework

1. Simulated Annealing with T=0 and omitting termination test

```
function SimulatedAnnealing(problem) returns solution state:

      current = node with problem initial state

      for t = 1 to infinity do:
            T = 0
            next = random successor for current
            Delta E = next.Value – current.Value
            if ΔE > 0 then current = next
            else current = next with probability e^(ΔE/T) ~ 1
            // current will always be next since T = 0
```

2. Genetic Algorithm with population of 1

```
population = set with population size 1
function GeneticAlgorithm(population, FITNESS-FN) returns individual:
      new_population =  empty set
      for i = 1 to SIZE(population) do: // loops only once since SIZE is 1
            x = RANDOM-SELECTION(population, FITNESS-FN)
            y = RANDOM-SELECTION(population, FITNESS-FN) // x and y are the same

            child = REPRODUCE(x,y)
            //reproduce with own genetic information, returns the same info
            if (small random probability) then child = MUTATE(child)
            add child to new_population
      population = new_population

return best individual in the population according to FITNESS-FN
// this will either be the same node (since parent and child have the same info after
reproducing with self, or the child if random mutation occurs and increases its
fitness
```

3. Hill Climbing with random restarts

```
function RandomRestartHillClimbing(problem) returns state of the best maximum:

	start = CHOSE-RANDOM(problem) // randomly chosen starting node
	best_run = current highest value state

	loop do:
		local_maximum = HillClimbing(start)
		if local_maximum.VALUE > best_run.VALUE
		then best_run = local_maximum

	until sufficient time has passed or a goal value is reached

	return best_run

function HillClimbing(problem) returns a state that is a local maximum:

	current = MAKE-NODE(problem.INITIALSTATE)
	loop do:
		neighbor = highest value successor of current
		if neighbor.VALUE <= current.VALUE then return current.STATE
		current = neighbor
```