Draft Solutions

In the solutions below "A" refers to Version A of the test; "B" refers to Version B of the test.

State Space Search (20 points)

greedy best-first (5 pts.)

(1) $A - B - D - J$

(2) $A - B - D - J$

iterative deepening (5 pts.)

(1) $(A-)A - B - C$
$A - B - D - E - F - C - G - H$
$A - B - D - J$

(2) $A - B - D - J$

uniform-cost (5 pts.)

(1) $A - C - B - H - F - E - G - D - 0$
$-K - M - S - J$

(2) $A - B - D - J$

A* (5 pts.)

(1) $A - C - H - G - B - E - F - D - K - J$

(2) $A - B - D - J$

Local Search (20 points)

1. The main disadvantage of hill-climbing is getting stuck at a local minimum/maximum. One method for alleviating this is to conduct a series of hill-climbing searches from randomly generated initial states, running each until it halts or makes no discernible progress, and returning the best of the results from each of the individual searches. Another solution is random-restart hill-climbing which restarts the hill-climbing search at a new random state after a pre-defined number of local steps. (Other answers are possible here.)

2. Simulated annealing allows steps *away* from the goal. As time passes, the probability of the downhill step decreases and the size of the downhill step decreases.

3. heuristic search and local search

4. **GA**$(Fitness, Fitness\_threshold, p, r, m)$

   - $P \leftarrow$ randomly generate p individuals
   - For each i in P, compute $Fitness(i)$

- While $[\max_i Fitness(i)] < Fitness\_threshold$
  1. Probabilistically **select** $(1-r)p$ members of P to add to P_S.
  2. Probabilistically choose $\frac{r \cdot p}{2}$ pairs of individuals from P. For each pair, $\langle i_1, i_2 \rangle$, apply **crossover** and add the offspring to P_s
  3. **Mutate** $m \cdot p$ random members of P_s
  4. $P \leftarrow P_s$
  5. For each i in P, compute $Fitness(i)$
- Return the individual in P with the highest fitness.

Adversarial Search (20 pts)

1. The values are:
   $N = 7,\ G = 3,\ D = 10,\ B = 10,\ J = 0,\ K = 11,\ F = 11,\ C = 11,\ A = 11$
   The player should take move C.
2. The first example of alpha-beta pruning shows allows nodes U and V to be pruned.
3. Yes.
4. On the order of b^d.
5. On the order of $b^{d/2}$. The branching factor at maximizing levels is b; but at minimizing levels, it is 1.

First-Order Logic and Resolution Theorem-Proving (20 points total)

1. (a) Every dog owner is an animal lover. $\forall x\ \ (\exists y\ \ Dog(y) \wedge Owns(x,y)) \rightarrow AnimalLover(x)$
   (b) Jack owns a dog. $\exists x : Dog(x) \wedge Owns(Jack, x)$
   (c) No animal lover kills an animal. $\forall x\ \ AnimalLover(x) \rightarrow \forall y\ \ Animal(y) \rightarrow \neg Kills(x,y)$
   (d) Tuna is a cat. $Cat(Tuna)$
   (e) Either Jack or Curiosity kills Tuna. $Kills(Jack, Tuna) \vee Kills(Curiosity, Tuna)$
   (f) Cats are animals. $\forall x\ \ Cat(x) \rightarrow Animal(x)$
2. - (1) $\neg Dog(S(x)) \vee \neg Owns(x, S(x)) \vee AnimalLover(x)$ (provided)
   - (2a) $Dog(D)$ (D is the function that finds Jack's dog)
     (2b) $Owns(Jack, D)$
   - (3) $\neg Dog(S(x)) \vee \neg Owns(x, S(x)) \vee AnimalLover(x)$ (provided)
   - (4) $Cat(Tuna)$
   - (5) $Kills(Jack, Tuna) \vee Kills(Curiosity, Tuna)$
   - (6) $\neg Cat(z) \vee Animal(z)$
3. (a) Add $\neg Animal(Tuna)$ to the KB as statement 7.
   (b) Resolve statements 7 and 6 (unifier: $z = Tuna$) to get $\neg Cat(Tuna)$ (which becomes statement 8).
   (c) Resolve statements 8 and 4 to produce *nil/fail.*