1. Give for each of the following four classes of games one example:

(a) deterministic, full-information [1] Othello

(b) deterministic, partial-information [1] Stratego

(c) stochastic, full-information [1] Backgammon

(d) stochastic, partial-information [1] Multiple hands of poker

2. Explain briefly how Particle Swarm Optimization works. [5]

PSO is a stochastic population-based search algorithm inspired by, for instance, the way birds flock. It searches the space of the objective function in an attempt to find the global optimum. Each individual search point has a location, velocity and memory of the best location it has found so far as well as the best location the population (global version) or its neighbors (local version) have found; the latter comprising the sole method of communication between individuals. Search progresses by moving towards a new location using a modified velocity based on a stochastic weighted combination of an individual’s own best location found and the global/local best location found.

3. Explain briefly how the local search technique Random- Restart Hill Climbing works. [2]

Random-Restart Hill Climbing conducts a series of hill-climbing searches from randomly generated initial states, stopping when a goal is found.

4. What is the diameter of this state space? Explain your answer! [1]

3; this is the largest minimal number of steps between any two nodes.

5. Give the execution trace for Uniform Cost Graph Search (UCGS). [8]

```
<table>
<thead>
<tr>
<th>open</th>
<th>closed</th>
<th>eval</th>
</tr>
</thead>
<tbody>
<tr>
<td>A0</td>
<td>-</td>
<td>A</td>
</tr>
<tr>
<td>B4E6</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>E6C6F6</td>
<td>AB</td>
<td>E</td>
</tr>
<tr>
<td>C6F6I6D9</td>
<td>ABE</td>
<td>C</td>
</tr>
<tr>
<td>F6D6G7H7I9</td>
<td>ABEC</td>
<td>F</td>
</tr>
<tr>
<td>D7G7H7I9</td>
<td>ABECF</td>
<td>D</td>
</tr>
<tr>
<td>G7H7I9</td>
<td>ABECFD</td>
<td>G</td>
</tr>
<tr>
<td>H7I9</td>
<td>ABECFDG</td>
<td>H</td>
</tr>
<tr>
<td>I8</td>
<td>ABECFDGH</td>
<td>I</td>
</tr>
</tbody>
</table>
```
goal found; solution = ABFHI; path-cost(ABFHI)=8
6. Is UCGS optimal for this problem? Explain your answer! [2]
   Yes, because the branching factor is finite, the step costs are all positive, and UCTS is always optimal under those conditions.

7. Give the execution trace of Iterative Deepening Depth First Graph Search (ID-DFGS). [10]
   \( \text{depth-limit}=0 \)
   \[
   \begin{array}{c|c|c}
   \text{open} & \text{closed} & \text{eval} \\
   \hline
   A & - & A \\
   \end{array}
   \]
depth-limit reached and no goal found
   \( \text{depth-limit}=1 \)
   \[
   \begin{array}{c|c|c}
   \text{open} & \text{closed} & \text{eval} \\
   \hline
   A & - & A \\
   BE & A & B \\
   E & AB & E \\
   \end{array}
   \]
depth-limit reached and no goal found
   \( \text{depth-limit}=2 \)
   \[
   \begin{array}{c|c|c}
   \text{open} & \text{closed} & \text{eval} \\
   \hline
   A & - & A \\
   BE & A & B \\
   CFE & AB & C \\
   FE & ABC & F \\
   E & ABCFE & E \\
   ID & ABCFE & I \\
   \end{array}
   \]
goal found; solution=AEI; path-cost(AEI)=9

8. Is ID-DFGS optimal for this problem? Explain your answer! [1]
   No, because UCGS found a shorter path.

9. Give the execution trace for A∗ Graph Search (A∗GS) employing heuristic \( h \). [6]
   \[
   \begin{array}{c|c|c}
   \text{open} & \text{closed} & \text{eval} \\
   \hline
   A3 & - & A \\
   B5E9 & A & B \\
   F8E9C10 & AB & F \\
   H8E9C10G10 & ABF & H \\
   I8E9C10G10D11 & ABFH & I \\
   \end{array}
   \]
goal found; solution=ABFHI; path-cost(ABFHI)=8

10. Is for this problem \( h \) admissible? Explain your answer! [4]
    Yes, because as can be seen from the following table no node overestimates the remaining path-cost:
    \[
    \begin{array}{c|c|c}
    \text{node} & h(\text{node}) & C^*(\text{node}) \\
    \hline
    A & 3 & 8 \\
    B & 1 & 4 \\
    C & 4 & 4 \\
    D & 2 & 3 \\
    E & 3 & 3 \\
    F & 2 & 2 \\
    G & 2 & 3 \\
    H & 1 & 1 \\
    I & 0 & 0 \\
    \end{array}
    \]

11. Is for this problem \( h \) consistent? Explain your answer! [2]
    No, because \( h(C) = 4 > 3 = c(C, D) + h(D) \).

12. Is A∗GS employing heuristic \( h \) optimal for this problem? Explain your answer! [1]
    Yes, because it found the same solution as UCGS which we previously stated to be optimal for this problem.
The next three questions are about the following adversarial “chance” tree.

13. Calculate the EXPECTIMINIMAX values for nodes B, C and D in the above adversarial “chance” tree. Show your calculations! [3]
   
   \[ \text{EXPECTIMINIMAX}(B) = 0.1 \times 2 + 0.9 \times 1 = 0.2 + 0.9 = 1.1 \]
   \[ \text{EXPECTIMINIMAX}(C) = 0.2 \times 3 + 0.2 \times 2 + 0.6 \times 5 = 0.6 + 0.4 + 3 = 4 \]
   \[ \text{EXPECTIMINIMAX}(D) = 0.7 \times 1 + 0.3 \times 8 = 0.7 + 2.4 = 3.1 \]

14. Which action will MAX choose, \(a_1\), \(a_2\), or \(a_3\)? Explain your answer! [1]

   \[ \text{MAX will choose action } a_2 \text{ because it has the highest EXPECTIMINIMAX value.} \]

15. If the utility values given for MIN were multiplied with a positive constant \(c\), which action would MAX then choose? Explain your answer! [2]

   \[ \text{MAX would still choose action } a_2 \text{ because multiplying with a positive constant is a positive linear transformation and such transformations do not change decisions made on the basis of EXPECTIMINIMAX values.} \]
The final questions are about the following adversarial search tree. State evaluation heuristic values for the max player are provided in the form of numbers following the letter labels of the states (e.g., A5 indicates that the heuristic value of state A for the max player is 5). The order in which successors are generated is from left to right. Example: A generates first B, then C, and finally D. Non-quiescent states are indicated by bold circled states.

A pre-initialized history table is provided as follows: all entries are zero except for those specified in the following table:

<table>
<thead>
<tr>
<th>Move</th>
<th>AC</th>
<th>AD</th>
<th>BF</th>
<th>BG</th>
</tr>
</thead>
<tbody>
<tr>
<td>HT value</td>
<td>5</td>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

16. Give the execution trace for HTQSABDLM(A,3,2,−∞,∞). [20] (Note: that’s DLM, not IDM!)

#define DLM( ) HTQSABDLM( ), #define Max( ) HTQSABMaxV( ), #define Min( ) HTQSABMinV( )

```
call   open   eval   value
DLM(A,3,2,−∞,∞) C5D2B0 C Min(C,2,2,−∞,∞)=6 6,∞ AC,6
     D2B0   D Min(D,2,2,6,∞)=3 6,∞ AC,6
     B0     B Min(B,2,2,6,∞)=2 6,∞ [AC,6]
Min(C,2,2,−∞,∞) H0I0 H Max(H,1,2,−∞,∞)=6 (SSS) −∞,6 CH,6 [HQ:1]
     I0     I Max(I,1,2,−∞,6)=9 (SSS,QS,prune) −∞,6 CH,6 [CH:6,IR:1,
Min(D,2,2,6,∞) J0K0 J Max(J,1,2,6,∞)=3 (prune) 6,∞ DJ,3 [DJ:1]
Max(J,1,2,6,∞) S0T0 S Min(S,0,2,6,∞)=1 (QS,SQS,prune) 6,∞ JS,1 [S-AD:1,AD-AO:1]
     T0     T Min(T,0,2,6,∞)=3 6,∞ JT,3 [JT:1]
Min(B,2,2,6,∞) F3G1E0 F Max(F,1,2,6,∞)=8 6,8 BF,8
     G1E0   G Max(G,1,2,6,8)=2 (SSS,QS,prune) 6,8 BG,2 [BG:2,GP:1,
PAA:1,AA-AL:1]
Max(F,1,2,6,∞) N0O0 N Min(N,0,2,6,∞)=3 6,∞ FN,3
     O0     O Min(O,0,2,6,∞)=8 8,∞ FO,8 [FO:1]
```

17. Which nodes, if any, get pruned by HTQSABDLM(A,3,2,−∞,∞)? [4]

K,U,AF,V,E,L,M,X,Al

18. What is the Principal Variant (PV) found by HTQSABDLM(A,3,2,−∞,∞)? [3]

A→C→H→Q