JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

TEST -1EXAMINATION- 2025

B.Tech-IV Semester (CSE)

COURSE CODE (CREDITS): 24B11CI411 (3) MAX. MARKS: 15

COURSE NAME: ARTIFICIAL INTELLIGENCE: RECENT TRENDS AND APPLICATIONS

COURSE INSTRUCTORS: AAYUSH SHARMA MAX. TIME: 1 Hour

Note: *(a)All questions are compulsory.*

(b) The candidate is allowed to make Suitable numeric assumptions wherever required for solving problems

<table>
<tr><th>Q.No</th><th>Question</th><th>CO</th><th>Marks</th></tr>
<tr><td>Q1</td><td>A hospital deploys an AI system to prioritize emergency patients. One day, it must choose between treating a young patient with low survival chances or an elderly patient with higher survival chances. How does AI make such decisions, and how does it compare to human ethical reasoning?</td><td>[CO 1]</td><td>3</td></tr>
<tr><td>Q2</td><td>A rescue robot is deployed in a collapsed building to find survivors. The robot can either use Depth-First Search (DFS) or Breadth-First Search (BFS) to explore rooms. Given that some paths may lead to dead ends and time is critical, which algorithm would be more suitable in each case and why?
Note that:
R = Robot initial position, F = Fire, V = Victim Position
Action sequence of the robot: Left, Right, Down, Up
Case A
<table>
<tr><td>R</td><td></td><td></td><td>F</td></tr>
<tr><td></td><td>F</td><td></td><td></td></tr>
<tr><td></td><td>F</td><td></td><td></td></tr>
<tr><td></td><td></td><td>V</td><td></td></tr>
</table>
Case B
<table>
<tr><td></td><td>V</td><td></td><td></td></tr>
<tr><td></td><td>F</td><td>F</td><td></td></tr>
<tr><td>R</td><td></td><td>F</td><td></td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
</table>
</td><td>[CO 1]
[CO 2]</td><td>3</td></tr>
<tr><td>Q3</td><td>An AI-powered maze solver is tested on two different heuristics:
• h_1 = Euclidean Distance to the goal
• h_2 = Manhattan Distance to the goal
The following data is collected:
<table>
<tr><th>Heuristic</th><th>Nodes Expanded</th><th>Total Path Cost</th><th>Execution Time (ms)</th></tr>
<tr><td>H1</td><td>85</td><td>50</td><td>120</td></tr>
<tr><td>H2</td><td>95</td><td>48</td><td>110</td></tr>
</table>
Answer the following:
1. Which heuristic is admissible (if any)?
2. Which heuristic performs better overall, considering path cost vs. computational efficiency?
3. If the maze had diagonal movement allowed, which heuristic would be more appropriate?</td><td>[CO 1]
[CO 2]</td><td>3</td></tr>
<tr><td>Q4</td><td>Write the pseudo code for the following problem:
Given a string s consisting of lowercase and/or uppercase letters, the task is to return the length of the longest palindrome that can be built with those letters. Letters are</td><td>[CO 2]</td><td>3</td></tr>
</table>

	case-sensitive, so "Aa" is not considered a palindrome. For example, if the input is s = "abccccdd", the output will be 7, as the longest palindrome that can be formed is "dccaccd". In another example, if the input is s = "a", the output will be 1, as the longest palindrome is the single character "a".		
Q5	What is the output of the following code:	[CO 2]	3

```
defalpha_beta(node, depth, alpha, beta, maximizing_player):
  global nodes_visited
nodes_visited += 1

  if depth == 0:
    return node['value']

  if maximizing_player:
    value = -float('inf')
    for child in node['children']:
      value = max(value, alpha_beta(child, depth-1, alpha, beta, False))
      alpha = max(alpha, value)
      if alpha >= beta:
        break
    return value
  else:
    value = float('inf')
    for child in node['children']:
      value = min(value, alpha_beta(child, depth-1, alpha, beta, True))
      beta = min(beta, value)
      if alpha >= beta:
        break
    return value

tree = {
  'children': [
   {
      'children': [
        {'children': [{'value': 5}, {'value': 8}, {'value': 3}]},
        {'children': [{'value': 2}, {'value': 7}, {'value': 1}]},
        {'children': [{'value': 6}, {'value': 4}, {'value': 9}]}
      ]
    },
  {
      'children': [
        {'children': [{'value': 15}, {'value': 2}, {'value': 10}]},
        {'children': [{'value': 4}, {'value': 5}, {'value': 6}]},
        {'children': [{'value': 7}, {'value': 8}, {'value': 9}]}
      ]
    },
   {
      'children': [
        {'children': [{'value': 1}, {'value': 12}, {'value': 11}]},
        {'children': [{'value': 3}, {'value': 2}, {'value': 6}]},
        {'children': [{'value': 7}, {'value': 5}, {'value': 4}]}
      ]
    }
  ]
}

nodes_visited = 0
result = alpha_beta(tree, 3, -float('inf'), float('inf'), True)
print(f"Final value: {result}, Nodes visited: {nodes_visited}")
```