VAISHNO COLLEGE OF ENGINEERING

Affliated to HPTU, Hamirpur and approved by AICTE

AI lab

Lab Manual

CSPC-414P (NEP Syllabus)

Department of Computer Science & Engineering

VillThapkour, PO Bhardoya, Tehsil Indora,Distt. Kangra (HP)-176403

Contact: 094183-18394, Web: www.vaishno.edu.in

Vision of Institute

To emerge as an institute of eminence in the fields of engineering, technology and management in serving the industry and the nation by empowering students with a high degree of technical managerial and practical competence.

Mission of Institute

M1 To strengthen the theoretical, practical and ethical dimensions of the learning process by fostering a cultural of research and innovation among faculty members and students.

M2 To encourage long term interaction between academia and industry through the involvement of industry for hands on implementation of the curriculum.

M3 To strengthen and molding students in professional ethical, social and environmental dimensions by encouraging participation in co-curricular extracurricular and CSR activities.

Vision of the Department

To emerge as a department of eminence in computer science and engineering in serving the industry and the nation by empowering students with high degree of technical and practical competence.

Mission of the department

M1 To strengthen the theoretical and practical aspects of learning process by strongly encouraging a computer cultural of research, innovation and hands on learning in computer science and engineering

M2 To encourage long term interaction between the department and IT industry, through the involvement of IT industry for hands on implementation of course curriculum.

M3 To widen the awareness of students in professional, ethical, social and environmental dimensions by encouraging their participation in co-curricular extracurricular and CSR activities.

Program Educational Objectives (PEOs) of the department

PEO 1: Engage in successful careers in industry, academia, and public service, by applying the acquired knowledge of Science, Mathematics and Engineering, providing technical leadership for their business, profession and community

PEO 2: Establish themselves as entrepreneur, work in research and development organization and pursue higher education

PEO 3: Exhibit commitment and engage in lifelong learning for enhancing their professional and personal capabilities.

PROGRAM OUTCOMES

PO1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2: Problem analysis: Identify, formulate, research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9: Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able tocomprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11: Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO 12: Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Program Specific Outcome (PSOs)

PSO1: Apply knowledge of mathematics, engineering sciences and multidisciplinary knowledge to the solution of computer science engineering problems.

PSO2: Apply research-based knowledge, appropriate techniques, IT tools to complex computer science engineering problems including design, analysis, interpretation of data, and synthesis of the information to provide valid conclusions.

PSO3: Apply ethical principles engineering profession and recognize the need of independent and lifelong learning for professional development and personnel growth.

Lab Syllabus & List of Experiments

CS-414P AI Lab							
Teaching Scheme			**Credit**	**Marks Distribution**			**Duration End Semester Examination**
L	**T**	**P/D**	**C**	**Internal Assessment**	**End Semester Examination**	**Total**	
0	**0**	**2**	**1**	**Maximum Marks: 30**	**Maximum Marks: 20**	**50**	**2hrs**
				Minimum Marks: 12	**Minimum Marks: 08**	**20**	

Following is the list of experiments/ jobs. Minimum 08 number of practicals are to be performed from following list. The additional experiments may be performed by the respective institution depending on the infrastructure available.

Laboratory Work:

1. Write a program to implement breadth first search algorithm.
2. Write a program to implement depth first search algorithm.
3. Write a program to implement the Hill climbing algorithm
4. Write a program to build and display Neural network using Tenser flow keres.
5. Write a program to implement Genetic algorithm.
6. Study of expert system tools and applications.
7. Write a program to implement Traveling salesman problem.
8. Write a program to implement four queen problem.
9. Write a program to solve monkey banana problem.
10. Write a program to implement Tower of Hanoi.

Evaluation Scheme

Internal Assessment: 30 marks (pass marks:12)

Distribution of marks for internal assessment:

- Written/presentation/Demonstration: 05
- Viva-voice: 05
- Teacher assessment: Lab Work performance/Report/File Work:15
- Attendance: 05

External Assessment: 20 marks (pass marks: 08)

Total marks 30+20=50, Pass marks = 20

Note: Student has to pass internal & external assessment separately.

GENERAL GUIDELINES AND SAFETY INSTRUCTIONS

1. You may use the computers in the lab only when a teacher is present.
2. Please place your bags at the front of the lab.
3. Do not eat or drink in the lab.
4. Keep the lab clean and neat at all times.
5. Use only the computer you are assigned to.
6. Report any hardware fault immediately to your teacher. Never attempt to dismantle the different parts of the computer.
7. Each student must log in to his/her account. No sharing of accounts is permitted.
8. The computers are for your academic use. Playing computer games for entertainment is strictly not allowed.
9. Shut down the computer properly after use.
10. Do not charge your personal mobile devices in the lab.

Cleanliness

- Keep your workspace clean and free of clutter
- Don't eat or drink in the lab
- Don't litter
- Don't remove cables or items from the lab

Fire safety

- Have a fire extinguisher and first-aid kit available
- Follow fire safety guidelines
- Be aware of the possibility of an accidental fire
- Know how to react to a fire
- Have a planned fire escape route

Eye and body safety

- Avoid eye fatigue by blinking often or closing your eyes for a few minutes
- Sit straight and in a comfortable posture
- Spread your fingers apart or rotate your wrists at regular intervals
- Wear proper lab attire
- Practice good hygiene

Other safety guidelines

- Don't spill liquids on the computer
- Don't touch hot or high voltage areas of printers

- Don't open a power supply or CRT monitor
- Don't tamper with wires or network cables
- Don't use illegal software
- Don't attempt to compromise network security

Practical No: 1

Aim:- Write a program to implement breadth first search algorithm.

Software Used:- Visual Studio Code

```
#include <iostream>
#include <queue>
#include <vector>
using namespace std;

// BFS from given source s
void bfs(vector<vector<int>>& adj, int s)
{
    // Create a queue for BFS
    queue<int> q;
```

```
// Initially mark all the vertices as not visited
// When we push a vertex into the q, we mark it as
// visited
vector<bool> visited(adj.size(), false);

// Mark the source node as visited and
// enqueue it
visited[s] = true;
q.push(s);

// Iterate over the queue
while (!q.empty()) {

    // Dequeue a vertex from queue and print it
    int curr = q.front();
    q.pop();
    cout << curr << " ";

    // Get all adjacent vertices of the dequeued
    // vertex curr If an adjacent has not been
    // visited, mark it visited and enqueue it
    for (int x : adj[curr]) {
        if (!visited[x]) {
            visited[x] = true;
```

```
                q.push(x);
            }
        }
    }
}

// Function to add an edge to the graph
void addEdge(vector<vector<int>>& adj,
                    int u, int v)
{
    adj[u].push_back(v);
    adj[v].push_back(u); // Undirected Graph
}

int main()
{
    // Number of vertices in the graph
    int V = 5;

    // Adjacency list representation of the graph
    vector<vector<int>> adj(V);

    // Add edges to the graph
    addEdge(adj, 0, 1);
    addEdge(adj, 0, 2);
```

```
    addEdge(adj, 1, 3);
    addEdge(adj, 1, 4);
    addEdge(adj, 2, 4);

    // Perform BFS traversal starting from vertex 0
    cout << "BFS starting from 0 : \n";
    bfs(adj, 0);

    return 0;
}
```

Practical No: 2

Aim:- Write a program to implement depth first search algorithm.

Software Used:- Visual Studio Code

```
#include <bits/stdc++.h>
using namespace std;

// Recursive function for DFS traversal
void DFSRec(vector<vector<int>> &adj, vector<bool> &visited, int s){

    visited[s] = true;

    // Print the current vertex
    cout << s << " ";

    // Recursively visit all adjacent vertices
```

```
    // that are not visited yet
    for (int i : adj[s])
        if (visited[i] == false)
            DFSRec(adj, visited, i);
}

// Main DFS function that initializes the visited array
// and call DFSRec
void DFS(vector<vector<int>> &adj, int s){
    vector<bool> visited(adj.size(), false);
    DFSRec(adj, visited, s);
}

// To add an edge in an undirected graph
void addEdge(vector<vector<int>> &adj, int s, int t){
    adj[s].push_back(t);
    adj[t].push_back(s);
}

int main(){
    int V = 5;
    vector<vector<int>> adj(V);

    // Add edges
    vector<vector<int>> edges={{1, 2},{1, 0},{2, 0},{2, 3},{2, 4}};
```

```
    for (auto &e : edges)
        addEdge(adj, e[0], e[1]);

    int s = 1; // Starting vertex for DFS
    cout << "DFS from source: " << s << endl;
    DFS(adj, s); // Perform DFS starting from the source vertex

    return 0;
}
```

Practical No: 3

Aim:- Write a program to implement Genetic algorithm.

Software Used:- Visual Studio Code

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <cstdlib>
#include <ctime>
using namespace std;

#define POP_SIZE 6      // Population size
#define CHROMO_LENGTH 5 // Length of binary chromosome
#define GENERATIONS 10  // Number of generations
#define MUTATION_RATE 0.1 // Mutation probability

// Function to evaluate fitness: f(x) = x^2
```

```
int fitness(int x) {
    return x * x;
}

// Convert binary chromosome to integer
int binaryToDecimal(vector<int>& chromo) {
    int value = 0;
    for (int bit : chromo) {
        value = (value << 1) | bit;
    }
    return value;
}

// Generate random chromosome
vector<int> generateChromosome() {
    vector<int> chromo(CHROMO_LENGTH);
    for (int i = 0; i < CHROMO_LENGTH; i++) {
        chromo[i] = rand() % 2;
    }
    return chromo;
}

// Perform single-point crossover
pair<vector<int>, vector<int>> crossover(vector<int>& parent1, vector<int>& parent2) {
    int point = rand() % (CHROMO_LENGTH - 1) + 1; // Avoid 0 or full swap
```

```
    vector<int> child1 = parent1;
    vector<int> child2 = parent2;
    for (int i = point; i < CHROMO_LENGTH; i++) {
        swap(child1[i], child2[i]);
    }
    return {child1, child2};
}

// Perform mutation
void mutate(vector<int>& chromo) {
    for (int i = 0; i < CHROMO_LENGTH; i++) {
        if ((rand() % 100) < (MUTATION_RATE * 100)) {
            chromo[i] = !chromo[i];
        }
    }
}

int main() {
    srand(time(0));
    vector<vector<int>> population(POP_SIZE);

    // Initialize population
    for (int i = 0; i < POP_SIZE; i++) {
        population[i] = generateChromosome();
    }
```

```
    for (int gen = 0; gen < GENERATIONS; gen++) {
        // Evaluate fitness
        vector<pair<int, vector<int>>> fitnessVals;
        for (auto& chromo : population) {
            int value = binaryToDecimal(chromo);
            fitnessVals.push_back({fitness(value), chromo});
        }

        // Sort by fitness (descending order)
        sort(fitnessVals.rbegin(), fitnessVals.rend());

        // Print best individual of the generation
        cout << "Generation " << gen << " Best: " << binaryToDecimal(fitnessVals[0].second) << "
Fitness: " << fitnessVals[0].first << endl;

        // Select top individuals for reproduction
        vector<vector<int>> new_population;
        for (int i = 0; i < POP_SIZE / 2; i++) {
            auto[parent1, parent2] = crossover(fitnessVals[i].second, fitnessVals[i + 1].second);
            mutate(parent1);
            mutate(parent2);
            new_population.push_back(parent1);
            new_population.push_back(parent2);
        }
```

```
        population = new_population;
    }


    return 0;
}
```

Practical No: 4

Aim:- Study of expert system tools and applications.

Software Used:- Visual Studio Code

1. Introduction to Expert Systems

An **Expert System (ES)** is an artificial intelligence-based computer system that mimics human decision-making capabilities by using knowledge and inference rules. These systems are widely used in problem-solving and decision-making tasks in various domains such as medicine, engineering, business, and more.

2. Components of an Expert System

1. **Knowledge Base**: Contains facts and heuristics (rules) to solve specific problems.
2. **Inference Engine**: Applies logical rules to the knowledge base to derive conclusions.
3. **User Interface**: Allows users to interact with the system.
4. **Explanation Module**: Provides explanations for decisions made by the system.
5. **Knowledge Acquisition Module**: Facilitates updating and refining the knowledge base.

---

3. Expert System Tools

Expert system tools are software frameworks used to develop expert systems. These tools provide environments for knowledge representation, reasoning, and decision-making.

A. Common Expert System Development Tools

1. **CLIPS (C Language Integrated Production System)**
    - Developed by NASA.
    - Efficient for rule-based expert systems.
    - Supports forward and backward chaining.

2. **Prolog (Programming in Logic)**
    - A logic-based programming language.
    - Used in AI applications requiring symbolic reasoning.
3. **MYCIN**
    - One of the earliest medical expert systems.

Practical No: 5

Aim:- Write a program to implement Traveling salesman problem.

Software Used:- Visual Studio Code

```
#include <iostream>

#include <vector>

#include <climits>

using namespace std;


#define N 4 // Number of cities


// Distance matrix representing the cost between cities

int dist[N][N] = {

   {0, 10, 15, 20},

   {10, 0, 35, 25},

   {15, 35, 0, 30},

   {20, 25, 30, 0}

};


bool visited[N];

int minCost = INT_MAX;
```

```
// Function to find the shortest path using backtracking
void tsp(int currentPos, int count, int cost, int start) {
    if (count == N && dist[currentPos][start]) { // If all cities visited
        minCost = min(minCost, cost + dist[currentPos][start]);
        return;
    }

    for (int i = 0; i < N; i++) {
        if (!visited[i] && dist[currentPos][i]) {
            visited[i] = true;
            tsp(i, count + 1, cost + dist[currentPos][i], start);
            visited[i] = false; // Backtrack
        }
    }
}

int main() {
    for (int i = 0; i < N; i++) visited[i] = false;
    visited[0] = true; // Start from the first city
    tsp(0, 1, 0, 0);

    cout << "The minimum cost of the TSP tour is: " << minCost << endl;
    return 0;
}
```

Practical No: 6

Aim:- Write a program to implement four queen problem.

Software Used:- Visual Studio Code

```
#include <iostream>
using namespace std;

#define N 4

// Function to print the solution
void printSolution(int board[N][N]) {
   for (int i = 0; i < N; i++) {
      for (int j = 0; j < N; j++) {
         cout << (board[i][j] ? "Q " : "- ");
      }
      cout << endl;
   }
   cout << endl;
}

// Function to check if a queen can be placed at board[row][col]
bool isSafe(int board[N][N], int row, int col) {
   for (int i = 0; i < col; i++) // Check left side
      if (board[row][i]) return false;

   for (int i = row, j = col; i >= 0 && j >= 0; i--, j--) // Check upper diagonal
      if (board[i][j]) return false;

   for (int i = row, j = col; i < N && j >= 0; i++, j--) // Check lower diagonal
      if (board[i][j]) return false;

   return true;
}

// Recursive function to solve the Four Queen problem
bool solveNQueens(int board[N][N], int col) {
   if (col >= N) { // All queens placed
      printSolution(board);
      return true; // Can be modified to return false if only one solution is needed
   }

   bool res = false;
   for (int i = 0; i < N; i++) {
      if (isSafe(board, i, col)) {
         board[i][col] = 1; // Place the queen
         res = solveNQueens(board, col + 1) || res;
         board[i][col] = 0; // Backtrack
      }
```

```
    }
    return res;
}

// Driver function
int main() {
    int board[N][N] = {0};
    if (!solveNQueens(board, 0)) {
        cout << "No solution exists" << endl;
    }
    return 0;
}
```

Practical No: 7

Aim:- Write a program to solve monkey banana problem.

Software Used:- Visual Studio Code

```
#include <iostream>

#include <queue>

#include <unordered_set>

using namespace std;


struct State {

    int monkey_pos;

    bool has_box;

    bool on_box;

    bool has_banana;


    bool operator==(const State& other) const {

        return monkey_pos == other.monkey_pos && has_box == other.has_box &&

               on_box == other.on_box && has_banana == other.has_banana;

    }
```

```
};

// Custom hash function for State
namespace std {
    template <>
    struct hash<State> {
        size_t operator()(const State& s) const {
            return hash<int>()(s.monkey_pos) ^ hash<bool>()(s.has_box) ^
                   hash<bool>()(s.on_box) ^ hash<bool>()(s.has_banana);
        }
    };
}

// Function to check if the monkey has reached the banana
bool isGoalState(const State& state) {
    return state.has_banana;
}

// Function to perform BFS to find the solution
void solveMonkeyBanana() {
    queue<State> q;
    unordered_set<State> visited;

    State initial = {0, false, false, false};
    q.push(initial);
```

```
visited.insert(initial);

while (!q.empty()) {
   State current = q.front();
   q.pop();

   if (isGoalState(current)) {
      cout << "Monkey has obtained the banana!" << endl;
      return;
   }

   // Possible actions
   if (!current.has_box) {
      State next = current;
      next.has_box = true;
      if (visited.find(next) == visited.end()) {
         q.push(next);
         visited.insert(next);
      }
   }

   if (current.has_box && !current.on_box) {
      State next = current;
      next.on_box = true;
      if (visited.find(next) == visited.end()) {
```

```
            q.push(next);
            visited.insert(next);
        }
    }

    if (current.on_box) {
        State next = current;
        next.has_banana = true;
        if (visited.find(next) == visited.end()) {
            q.push(next);
            visited.insert(next);
        }
    }
  }
  cout << "No solution found!" << endl;
}

int main() {
  solveMonkeyBanana();
  return 0;
}
```

Practical No: 8

Aim:- Write a program to implement Tower of Hanoi.

Software Used:- Visual Studio Code

```
#include <iostream>
using namespace std;
```

```
// Function to solve the Tower of Hanoi problem
void towerOfHanoi(int n, char source, char auxiliary, char destination) {
   if (n == 1) {
      cout << "Move disk 1 from " << source << " to " << destination << endl;
      return;
   }
   towerOfHanoi(n - 1, source, destination, auxiliary);
   cout << "Move disk " << n << " from " << source << " to " << destination << endl;
   towerOfHanoi(n - 1, auxiliary, source, destination);
}

int main() {
   int n;
   cout << "Enter the number of disks: ";
   cin >> n;
   towerOfHanoi(n, 'A', 'B', 'C');
   return 0;
}
```


Viva Questions

1. What do you mean by AI.
2. Breadth first search & depth first search.
3. What are heuristics search techniques.
4. What do you mean by intelligent systems.
5. What do you mean by cybernetics & brain simulation.
6. What are the impact of AI on jobs & society.
7. What do you mean by machine learning.
8. Applications of Machine learning.
9. What do you mean by data normalization.
10. What do you mean by Artificial neural network.
11. What do you mean by Activation functions
12. What do you mean by perceptron.
13. What do you mean by fuzzy logic.
14. What do by mean by Genetic algorithm.
15. What do you mean by mutation & off spring.

Laboratory Experiment Evaluation Rubric

Category	Outstanding (Up to 100%)	Accomplished (Up to 75%)	Developing (Up to 50%)	Beginner (Up to 25%)
Written/Presentation/Demonstration	The write-up is clear, well-organized, and follows the prescribed format. All required sections (aim, apparatus, theory, procedure, diagram, etc.) are present and well-written. Demonstration is clear and thorough.	The report follows the specified format, but some sections (like the diagram or theory) are missing or incomplete. The demonstration is understandable but lacks depth.	The report includes most sections but lacks clarity, coherence, or completeness in some parts (e.g., diagram missing, unclear theoretical explanation). The demonstration is incomplete or unclear.	The report is poorly written and organized. Many sections are missing or incorrect (e.g., no diagram, incomplete procedure). The demonstration lacks clarity or is missing.
Viva-Voice	Demonstrates a deep understanding of the experiment, underlying principles, and outcomes. Answers questions confidently and	Demonstrates a general understanding of the experiment and principles but struggles with some aspects. Provides correct answers to most	Struggles with some fundamental concepts and principles. Answering questions requires additional prompts, with a few errors in understandin	Lacks a basic understanding of the experiment. Unable to answer most questions accurately. Demonstrates significant gaps in knowledge.

Category	Outstanding (Up to 100%)	Accomplished (Up to 75%)	Developing (Up to 50%)	Beginner (Up to 25%)
	accurately.	questions.	g.	
Performance/Report/File Work	Performs the experiment accurately and efficiently. The report is thorough, with correct observations, calculations, and analysis. Data is recorded neatly and with appropriate units. All relevant calculations and interpretations are included.	Performs the experiment well with minor errors or delays. The report is complete but may contain some inaccuracies or missing components in calculations or observations.	Completes the experiment but with notable mistakes, either in the setup or the data. The report has several missing or inaccurate components, including incorrect or incomplete calculations.	Struggles to perform the experiment correctly. Significant errors in setup, data collection, and analysis. The report is poorly structured with major inaccuracies or missing sections.
Attendance	Consistently attends all lab sessions, actively participates, and engages with the experiment and group discussions.	Attends most lab sessions with occasional absences. Participation is generally good but lacks consistency or	Attends some lab sessions but has frequent absences or minimal participation.	Misses several lab sessions and shows minimal to no participation in class or group activities.

Category	Outstanding (Up to 100%)	Accomplished (Up to 75%)	Developing (Up to 50%)	Beginner (Up to 25%)
		depth.		