Picat Programming Questions

Instructions

Answer the following questions using the Picat programming language. Unless otherwise specified, you may use any language constructs available in Picat, such as recursion, for-each loops, list operations, but not assignments.

1. Functional Programming

F1. **sphere_volume(R)**: A function that computes the volume of a sphere, given its radius R.

F2. **fact(N)**: A recursive function that compute $N!$. For example, `fact(5)` should return 120.

F3. **sum(List)**: A function that computes the sum of all integers in a list using recursion.

F4. **number_of_zeros(Lst)**: A function that returns the number of zeros in a given simple list of numbers Lst.

F5. **fib(N)**: A function that returns the Nth Fibonacci number using recursion.

F6. **Map Function**: Implement a function `map(F, List)` that applies a unary function `F` to every element of `List` and returns the new list.

F7. **List Filtering**: Implement a function `filter(Pred, List)` that returns a new list containing only elements satisfying the predicate `Pred`.

2. Predicates and Backtracking

B1. **Member Predicate**: Implement `member(X, L)` to succeed if `X` is in list `L`.

B2. **List Permutations**: Write a program that generates all permutations of a given list using backtracking.

B3. **N-Queens Problem**: Write a program to place N queens on an $N \times N$ chessboard so that no two queens attack each other.

B4. **Subset Generation**: Write a Picat program that generates all subsets of a given set using backtracking.

B5. **Simple Sudoku Solver**: Write a backtracking solver for a 4×4 Sudoku puzzle without using constraints.

3. Constraint Satisfaction and Optimization

C1. Magic Square: Use the `cp` module to generate a 3×3 magic square where each row, column, and diagonal sums to the same value.

C2. SEND+MORE=MONEY Puzzle: Solve the cryptarithmetic puzzle `SEND + MORE = MONEY` using constraint programming.

C3. Job Scheduling: Assign 3 jobs to 3 workers with given costs to minimize the total cost using `cp`.

C4. Graph Coloring: Color the vertices of a given graph using the fewest colors possible so that no two adjacent vertices share the same color.

C5. Knapsack Problem: Solve the 0/1 Knapsack problem using `cp` to maximize the value without exceeding the weight capacity.

4. Planning and Tabling

P1. Robot Path Planning: Use the `planner` module to plan a sequence of moves for a robot from a start position to a goal on a grid with obstacles.

P2. Missionaries and Cannibals: Model the classic Missionaries and Cannibals problem using `planner` to find a sequence of moves.

P3. Longest Common Subsequence: Use tabling to compute the length of the longest common subsequence of two strings.

P4. Coin Change Problem: Use tabling to find the minimum number of coins needed to make a given amount.

P5. Traveling Salesman Problem: Use tabling to find the shortest Hamiltonian cycle visiting all cities exactly once.