Name: Shabbar Adamjee
Roll No.: PB57
PRN: 1032221508

AIES ASSIGNMENT 3

CONSTRAINT SATISFACTION PROBLEM

Code

```cpp
#include <deque>
#include <iostream>
#include <map>
#include <set>
#include <string>

std::string send = "send";
std::string more = "more";
std::string money = "money";

bool checkSoln(const std::map<char, int> &letterMap) {
  std::string sendCopy = send;
  std::string moreCopy = more;
  std::string moneyCopy = money;

  for (auto &letter : sendCopy) {
    letter = '0' + letterMap.at(letter);
  }

  for (auto &letter : moreCopy) {
    letter = '0' + letterMap.at(letter);
  }

  for (auto &letter : moneyCopy) {
    letter = '0' + letterMap.at(letter);
  }

  int send = std::stoi(sendCopy);
  int more = std::stoi(moreCopy);
  int money = std::stoi(moneyCopy);

  return (send + more == money);
}

bool CSP(std::set<char> &uniqueLetters, std::map<char, int> &letterMap,
```

```
        std::map<int, bool> &numMap, std::deque<char> &letterDQ) {
  if (letterDQ.empty()) {
    return checkSoln(letterMap);
  }

  char currentLetter = letterDQ.front();
  letterDQ.pop_front();

  for (int numToAssign = 0; numToAssign < 10; ++numToAssign) {
    if (!numMap[numToAssign]) {
      // Try assigning this number to the current letter
      letterMap[currentLetter] = numToAssign;
      numMap[numToAssign] = true;

      // Continue with the next letter
      if (CSP(uniqueLetters, letterMap, numMap, letterDQ)) {
        return true; // Solution found
      }

      // Backtrack: Unassign the number and try another
      letterMap.erase(currentLetter);
      numMap[numToAssign] = false;
    }
  }

  // Push the letter back into deque and backtrack
  letterDQ.push_front(currentLetter);
  return false;
}

int main() {
  // Get characters from the 3 strings
  std::set<char> uniqueLetters;

  for (char s : send) {
    uniqueLetters.insert(s);
  }
  for (char s : more) {
    uniqueLetters.insert(s);
  }
  for (char s : money) {
    uniqueLetters.insert(s);
  }

  char firstLetter = money[0];

  // Map the letters to their numbers
  std::map<char, int> letterMap;
```

```
  letterMap[firstLetter] = 1; // 'm' must be 1 because MONEY is 5 digits

  // Map whether a number is used or not
  std::map<int, bool> numMap;
  for (int i = 0; i < 10; ++i) {
    numMap[i] = false;
  }
  numMap[1] = true; // 'm' is already used as 1

  // Queue up letters to assign (excluding 'm' since it's fixed)
  std::deque<char> letterDQ;
  for (char letter : uniqueLetters) {
    if (letter != firstLetter) {
      letterDQ.push_back(letter);
    }
  }

  // Solve the CSP
  bool solved = CSP(uniqueLetters, letterMap, numMap, letterDQ);

  if (solved) {
    std::cout << "Solution found!" << std::endl;
    for (const auto &x : letterMap) {
      std::cout << x.first << " = " << x.second << std::endl;
    }
  } else {
    std::cout << "No solution found." << std::endl;
  }

  return 0;
}
```

Output

```
AIES .\csp.exe
Solution found!
d = 7
e = 5
m = 1
n = 6
o = 0
r = 8
s = 9
y = 2
```

Assignment No. 3

Page No
Date :

Title
Implementation of solution of Constraint Satisfaction Problem like
SEND + MORE = MONEY OR CROSS + ROADS = DANGER

FAQs

1. What are other CSPs?

- Sudoku - Filling a 9×9 grid with digits such ~~as~~ that no number repeats in any row, column, or sub-grid.

- N-Queens - Placing N queens on an N×N chessboard such that no 2 queens attack each other.

- Map Colouring - Assigning colours to a map's regions such that no adjacent regions share the same color.

- Job Scheduling - Assigning tasks to time slots or workers, subject to various constraints.

2. What is meant by constraint propagation?

Refers to the process of enforcing constraints early in the problem-solving process to reduce the search space. ~~Python~~

When a variable is assigned a value, this assignment can limit the possible values of other related variables. By propagating these constraints throughout the system, the algorithm can eliminate inconsistent values and make further decisions faster, increasing efficiency.

3. Why can backtracking search be used to solve CSPs?

Because it systematically explores the search space by assigning values to variables one at a time and checking if the assignment is consistent with the constraints.

If a conflict arises, the algorithm backtracks to the previous decision point and tries a different value. This approach ensures all potential solutions are considered while pruning inconsistent paths early, making it efficient for problems like cryptarithmetic puzzles.

2/2/19