|  |  |
| --- | --- |
| **Celebrity in C++** | |
| #include <iostream>  #include <stack>  using namespace std;  void findCelebrity(int arr[][4], int n) {  stack<int> st;  for (int i = 0; i < n; i++) {  st.push(i);  }  while (st.size() > 1) {  int i = st.top();  st.pop();  int j = st.top();  st.pop();  if (arr[i][j] == 1) {  st.push(j);  } else {  st.push(i);  }  }  int potential = st.top();  bool isCelebrity = true;  for (int i = 0; i < n; i++) {  if (i != potential) {  if (arr[i][potential] == 0 || arr[potential][i] == 1) {  isCelebrity = false;  break;  }  }  }  if (isCelebrity) {  cout << potential << endl;  } else {  cout << "none" << endl;  }  }  int main() {  // Hardcoded input  int n = 4;  int arr[4][4] = {  {0, 0, 0, 0},  {1, 0, 1, 1},  {1, 1, 0, 1},  {1, 1, 1, 0}  };  // Finding the celebrity  findCelebrity(arr, n);  return 0;  } | Each cell arr[i][j] tells us whether person i knows person j.  int arr[4][4] = {  {0, 0, 0, 0}, // Person 0 knows nobody  {1, 0, 1, 1}, // Person 1 knows 0, 2, 3  {1, 1, 0, 1}, // Person 2 knows 0, 1, 3  {1, 1, 1, 0} // Person 3 knows 0, 1, 2  }; 🧱 Stack-Based Elimination Table  | **Step** | **Stack Before** | **i (pop1)** | **j (pop2)** | **arr[i][j]** | **Action Taken** | **Stack After** | | --- | --- | --- | --- | --- | --- | --- | | 1 | [0, 1, 2, 3] | 3 | 2 | 1 | 3 knows 2 → eliminate 3 | [0, 1, 2] | | 2 | [0, 1, 2] | 2 | 1 | 1 | 2 knows 1 → eliminate 2 | [0, 1] | | 3 | [0, 1] | 1 | 0 | 1 | 1 knows 0 → eliminate 1 | [0] |   Now stack.top() gives us **potential celebrity = 0** 🔍 Verification Table Check if person 0 is a **celebrity**:   | **i** | **arr[i][0] (i knows 0)** | **arr[0][i] (0 knows i)** | **Condition Satisfied?** | | --- | --- | --- | --- | | 0 | — | — | Skip self | | 1 | 1 | 0 | ✅ Person 1 knows 0, 0 knows no one | | 2 | 1 | 0 | ✅ Person 2 knows 0 | | 3 | 1 | 0 | ✅ Person 3 knows 0 |   ✔️ All conditions met — 0 is a celebrity ✅ Final Output: 0 |
| 0 | |