CSCI 104L Lecture 23: Backtracking

In the n-Queens problem, you want to place n queens on an $n \times n$ chessboard, so that no two queens can attack each other (directly horizontal, vertical, or diagonally).

```
__Q_
_Q__
__Q_
Q___
```

Question 1. Why is this not a valid solution to the 4-Queens problem?

Question 2. How could you fix it?

The following is a generic outline for all backtracking solutions:

```
void search (int row) {
  if (row==n) printSolution(); //Show the layout
  else {
    for(q[row]=0; q[row] < n; q[row]++)
      search(row+1);
  }
}
```

Note that all backtracking solutions have:

- A function that will be called recursively (with a parameter to keep track of where you are)
- A base case
- A loop to try all possible choices, with a recursive function call inside.

Here is the complete recursive function:

```
int *q; //positions of the queens
int n; //size of the grid
int **t; //threatened squares
void search (int row) {
  if (row==n) printSolution(); //Show the layout
  else {
    for(q[row]=0; q[row] < n; q[row]++)
      if (t[row][q[row]] == 0) {
        changeThreats(row, q[row], 1);//queen placed here!
        search(row+1);
        changeThreats(row, q[row], -1);//queen removed from here!
      }
  }
}
```

Now we also have to implement changeThreats. Since we are only worrying about placing on the **rest** of the board, we will only track threats on rows after the current one:

```
void changeThreats(int row, int column, int change) {
  for (int j = row+1; j < n; j++) {
    t[j][column] += change;
    if (column+(j-row) < n) t[j][column+(j-row)] += change;
    if (column-(j-row) >= 0) t[j][column-(j-row)] += change;
  }
}
void printSolution() {
  for(int i = 0; i < n; i++) {
    for(int j = 0; j < n; j++) {
      if (j == q[i]) cout << "Q";
      else cout << "_";
    }
    cout << endl;
  }
}
int main(void) {
  cin >> n;
  q = new int[n];
  t = new int* [n];
  for (int i = 0; i < n; i++) {
    t[i] = new int[n];
    for (int j = 0; j < n; j++) t[i][j] = 0;
  }
  search(0);
  delete [] q;
  for (int i = 0; i < n; i++) delete [] t[i];
  delete [] t;
  return 0;
}
```