

TASK: 14 Practice problem for coding preparation

N QUEENS

Given a chess board having $N \times N$ cells, you need to place N queens on the board in such a way that no queen attacks any other queen.

INPUT: The only line of input consists of a single integer denoting N .

OUTPUT:

It is possible to place all the N queens in such a way that no queen attacks another queen, then print N lines having N integers. The integer in the i th line and j th column will denote the cell (i, j) of the board and should be 1 if a queen is placed at (i, j) otherwise. If there are more than one way of placing queens print any of them. If it is not possible to place all the N queens in the desired way, then print 'Not possible'.

CONSTRAINTS:

$1 \leq N \leq 10$

SAMPLE INPUT:

4

SAMPLE OUTPUT:

0 1 0 0
0 0 0 1
1 0 0 0
0 0 1 0

Algorithm:

- 1) Define a function to check if a queen can be placed in a given cell of the board.
- 2) Define a function to recursively place queens on the board.
- 3) In the recursive function, check if all the queens have been placed. If yes, return the board configuration.
- 4) If not, loop through all the cells in the current row and check if a queen can be placed in that cell using the `isSafe()` function.
- 5) If a queen can be placed in a cell, mark the cell as occupied and recursively call the function for the next row.
- 6) If no queen can be placed in any cell of the current row, return none.

2) In the main function call the recursive function to place the queens and print the board configuration.

Program:

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
#define MAX_N 10
```

```
int n;
```

```
int board[MAX_N][MAX_N];
```

```
int is_valid(int row, int col) {
```

```
    int i, j;
```

```
    for (i = 0; i < n; i++) {
```

```
        if (board[i][col]) return 0;
```

```
    return 1;
```

```
    for (i = row; i < n; i++) {
```

```
        if (board[i][col]) return 0;
```

```
    return 1;
```

```
    for (i = row; i < n; i++) {
```

```
        return 0;
```

```
    return 1;
```

```
0 0 1 0
```

```
1 0 0 0
```

```
0 0 0 1
```

```
0 1 0 0
```

```
for (row=0; row<n; row++) {
```

```
if (is-valid(row, col)) {
```

```
board[row][col] = 1;
```

```
if (solve(col+1))
```

```
return 1;
```

```
board[row][col] = 0;
```

```
}
```

```
} return 0;
```

```
} return 1;
```

```
}
```

```
int main()
```

```
{ int i, j;
```

```
scanf("%d", &n);
```

```
if (solve(0))
```

```
for (i=0; i<n; i++)
```

```
for (j=0; j<n; j++)
```

```
{
```

```
printf("%d", board[i][j]);
```

```
}
```

```
printf("\n");
```

```
}
```

```
} else {
```

```

print("Not possible")
}
return 0;
}

```

Output:

sample Input:

4

Sample Output:

0100

0001

1000

0010

The Castle Gate:

Audi, a fun loving girl from the city of DUN, travels to Azkhar - a strange land beyond the mountains. She arrives at the gates of castle grey, owned by Puchi, the Lord of Azkhar to claim the treasure that is guarded. However, the destiny has other plans for her as she has to move through floors crossing obstacles on her way to reach the treasure.

Algorithm:

Read the input value of t

Repeat the following steps for T test cases.

3. End.

Program:

#include <stdio.h>

int count(int n) {

int count = 0;

while (n > 0) {

n = (n - 1);

count++;

}

return count;

}

int main() {

int t, n;

scanf("%d", &t);

while (t-- > 0) {

scanf("%d", &n);

int count = 0;

for (int i = 1; i <= n; i++) {

for (int j = 1; j <= n; j++) {

if (i * j <= n) {

count++;

}

}

}

```
printf("%d\n", count);
}
return 0;
}
```

Output:

Sample Input: 3 4 6 8

Sample output:

3
12
21

VEL TECH - CSE	
EX NO.	4
PERFORMANCE (5)	4
RESULT AND ANALYSIS (3)	7
INVOICE (3)	7
RECORD (4)	
PG. NO. (15)	
SIGN WITH DATE	

Result: Thus the program is executed and verified successfully.