Task - 14(A)

Practice Problems for Coding Preparation.

N Queens

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

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 function, check if all
3. In the recursive function, check if all the queens on the 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 is safe() function.
5. If a queen can be queen can be placed in a cell, mark the cell as occupied and recursively call the function for the next row.
6. If the recursive call returns a board, return it.
7. unmark the cell and continue the loop for the next cell.
8. If no queen can be placed in any cell of the next row, return None.

```
#include <stdio.h>
int countSetBits(int n) { int count = 0;
while (n > 0) { n &= (n - 1); count++;
}
return count;
}
int main() { int t, n;
scanf("%d", &t);
while (t--) {
scanf("%d", &n);

int count = 0;
for (int i = 1; i <= n; i++) {
for (int j = i + 1; j <= n; j++) { if ((i ^ j) <= n) {
count++;
}
}
}
printf("%d\n", count);
}
return 0;
}
```

Task - 14(B)

The Castle Gate

Algorithm

1. Read the input value of T.
2. Repeat the following steps for T test cases:
   a. Read the value of N.
   b. Initialize the variable count to 0.
   c. Iterate i from 1 to N-1.
      1. Iterate j from i+1 to N.
   d. If (i XOR j) is less than or equal to N, increment count.
   e. Print count.
3. End.

```
#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[row][i] || board[i][col]) return 0;

}

for (i = row, j = col; i >= 0 && j >= 0; i--, j--) { if (board[i][j])

return 0;

}

for (i = row, j = col; i < n && j >= 0; i++, j--) { if (board[i][j])

return 0;

}

return 1;

}

int solve(int col) { int row, i;

if (col >= n) return 1;

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;
```

```
}
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 {
printf("Not possible\n");
}
return 0;
}
```

...EL TECH - C...B	
...NO.	14
...RFORMANCE (5)	5
...ULT AND ANALYSIS (3)	3
...A VOCE (3)	3
...CORD (4)	4
T... ...L (15)	15
SIG... ...WITH DATE	

Result :-

Thus, the program is verified and executed successfully