```
*********Program 43.2.c ******************************
//Recursive N queens ..
#include<stdio.h>
#include<stdlib.h>
#define not_attacked 0
#define TRUE 1
#define FALSE 0
#define N 4

int Queen_Status(int);
void Place_Queen(int);
void print_sol();
void printtabs(int);

int Q[N+1];

int main() {
  int i = 1;
  Place_Queen(i);
}

void Place_Queen(int i) {
  printtabs(i);
  printf("Placing Queen at col %d - \n",i);
  for (Q[i] = 1; Q[i] <= N; Q[i]++) {
    if (Queen_Status(i) == not_attacked) {
      if (i == N) {
        printtabs(i);
        printf("vacant slot - Q[%d] = %d AND sol. found\n", i, Q[i]);
        print_sol();
        exit(1); //stop as soon as a sol. is found.
      } else {
        printtabs(i);
        printf("vacant slot - Q[%d] = %d\n", i, Q[i]);
        printtabs(i);
        printf("calling Place_Queen(%d)\n", i+1);
        Place_Queen(i+1);
        printtabs(i);
        printf("returned after trying to place queen at col %d\n", i+1);
      }
    }
  }
  return; //control returns back to main or Place_Queen(i-1);
}

int Queen_Status(int i) {
  int j, attacked = FALSE;
  for (j = 1; j < i; j++) {
    if (Q[i] == Q[j] || abs(Q[i]-Q[j]) == i - j) {
      attacked = TRUE;
      return attacked;
    }
  }
  return attacked;
}

void print_sol() {
 int i;
 printf("\n**********************\n");
 for (i = 1; i <= N; i++) {
   printf("Q[%d] = %d ", i, Q[i]);
 }
```

```
 printf("\n**********************\n");
 return;
}

void printtabs(int i) {
  int k;
  i = i - 1;
  for (k = 0; k < i; k++) {
    printf("\t");
  }
}
```