```
*********Program 42.c ****************************
#include<stdio.h>
//recursive binary search
int binsearch(int a[], int key, int low, int high);
int main() {
  int a[7] = {10, 20, 30, 40, 50, 60, 70};
  int i, n = 7;
  int key, res;
  printf("enter the key: ");
  scanf("%d",&key);
  printf("Array is ");
  for (i = 0; i < 7; i++) {
    printf("%d ", a[i]);
  }
  printf("\n");
  res = binsearch(a, key, 0, n-1);
  if (res == -1) {
    printf("%d not found\n", key);
  } else {
    printf("%d key found in array element %d\n", key, res);
  }
}

int binsearch(int a[], int key, int low, int high) {
 int mid;
 printf("Searching from low: %d high: %d\n", low, high);
 if (high < low) {
   return -1; //element not found
 }
 mid = low + (high - low) / 2;
 if (a[mid] > key) {
   return binsearch(a, key, low, mid-1);
 } else if (a[mid] < key) {
   return binsearch(a, key, mid + 1, high);
 } else {
   return mid; //element found...
 }

}
*********Program 43.1.c ****************************
#include<stdio.h>
#include<stdlib.h>
//check whether Queens are safe or not..
int main() {
 int q[5] = {0, 2, 4, 1, 3};
 int i, j;
 for (i = 4;  i >= 1; i--) {
   for (j = 1; j < i; j++) {
     if (q[i] == q[j] || abs(q[i] - q[j]) == i - j) {
       printf("Queens are attacked\n");
       exit(1);
     }
   }
 }
 printf("Queens are safe\n");
}
*********Program 43.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();

int Q[N+1];

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

void Place_Queen(int i) {
  for (Q[i] = 1; Q[i] <= N; Q[i]++) {
    if (Queen_Status(i) == not_attacked) {
      if (i == N) {
        print_sol();
        exit(1); //stop as soon as a sol. is found.
      } else {
        Place_Queen(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;
 for (i = 1; i <= N; i++) {
   printf("Q[%d] = %d ", i, Q[i]);
 }
 printf("\n**********************\n");
 return;
}
```