1 tree_search2.c (print_data() function only)

```c
void print_data(data_el *cur)
{
    if (cur->left)
        print_data(cur->left);

    printf("%d ", cur->value);

    if (cur->right)
        print_data(cur->right);
}
```

Computation structures / concepts:

- Tree traversal using recursion.

2 tsp.c

```c
/*

tsp.c

This program is a solution to the traveling salesman problem, and illustrates recursion.

*/

#define N 5
#define INF 1000 /* larger than a trip could possibly cost */
#define TRUE 1
#define FALSE 0

int cost_matrix[N][N] = {
    { 0, 2, 3, 1, 7 },
```
\{ 2, 0, 5, 7, 4 \},
\{ 3, 5, 0, 6, 1 \},
\{ 1, 7, 6, 0, 3 \},
\{ 7, 4, 1, 3, 0 \}
};

int solution[N];
int solution_cost = INF;
void tsp(int *, int, int, int);
int not_visited(int *, int, int, int);

int main(int argc, char **argv) {
    int tour[N], i;

    tour[0] = 0; /* start at city 0 */
tsp(tour, 1, 0, N);

    /* print answer */
    printf("Best tour for %d cities:\n", N);
    for (i = 0; i < N; i++)
        printf("%d ", solution[i]);
    printf("\n");
    printf("Cost: %d\n", solution_cost);
}

/* recursive function that explores all tours */
void tsp(int *tour, int num_tour, int tour_cost, int num_cities)
{
    int i;

    /* base case */
    if (num_tour == num_cities) {
        printf("tour: ");
        for (i = 0; i < N; i++)
            printf("%d ", tour[i]);
        printf("\n");
        printf("cost: %d\n", tour_cost);

        if (tour_cost < solution_cost) {
            for (i = 0; i < N; i++) solution[i] = tour[i];
            solution_cost = tour_cost;
        }
        return;
    }

    /* recursive case */
for (i = 0; i < num_cities; i++) {
    if (not_visited(tour, num_tour, i)) {
        tour[num_tour] = i;
        tsp(tour, num_tour+1, tour_cost + cost_matrix[tour[num_tour-1]][i], num_cities);
    }
}

/* tests if a city has been visited in current partial tour */
int not_visited(int *tour, int num_tour, int city)
{
    int i;

    for (i = 0; i < num_tour; i++)
        if (tour[i] == city)
            return FALSE;

    return TRUE;
}

Computation structures / concepts:

- Recursion
- Introduction to class traveling salesman problem
- Partial tour expansion

3  hanoi.c

/*
   hanoi.c

   Implementation of the Tower of Hanoi problem. Code adapted from an example developed by Gang Qu.
*/

#include <stdio.h>

int counter;

void hanoi(char from, char to, char middle, int n)
{ 
    if (n == 1) 
        printf("%3d: Move disk from %c to %c\n", ++counter, from, to);
    else 
    { 
        hanoi(from, middle, to, n-1);
        hanoi(from, to, middle, 1);
        hanoi(middle, to, from, n-1);
    }
}

int main (void)
{
    int disks;

    printf("Enter the number of disks: ");
    scanf("%d", &disks);
    printf("\n\n");
    hanoi('A', 'C', 'B', disks);
    printf("\n\nTotal number of moves: %d\n", counter);
    return 0;
}

Computation structures / concepts:

- Recursion
- Introduction to classic Tower of Hanoi problem