tower.c

```
Includes

Prototypes

Int moves = 0

Main function
main(argc, **argv) {
    Char *validchars = “srn:”
    Disks = 5
    Loop to take in user input
    Getopt loop {
        Switch option {
        case stack
            Stack = true
        Case recursive
            Recursion = true
        Case num
            Number = optarg
        }
    }
    Function calls based on user input
    If stack {
        print(stack(number))
        Moves = 0
    }
    If recur {
        print(recursion(number, ‘A’, ‘B’, ‘C’))
        Moves = 0
    }
    Return 0
}

Stack solution
stack(num_of_disks) {
    Create 3 stacks (A, B, C) with capacity of num_of_disks
    For loop {
        Add a disk to stack A until its full
    }
    Do this loop until finished if the total number of disks is odd
    If (num_of_disks is odd) {
        while(true) {
```

```
        Make legal move between A and B, add 1 to moves, and check if done
        If ((A < B and A is not empty) OR if B is empty) { // A and B
            Move top disk of A to B
        }
        Else {
            Move top disk of B to A
        }
        moves++
        Break loop if A and C are both empty

        Make legal move between A and C, add 1 to moves, and check if done
        If ((A < C and A is not empty) OR if C is empty) { // A and C
            Move top disk of A to C
        }
        Else {
            Move top disk of C to A
        }
        moves++
        Break loop if A and C are both empty

        Make legal move between B and C, add 1 to moves, and check if done
        If ((B < C and B is not empty) OR if C is empty) { // B and C
            Move top disk of B to C
        }
        Else {
            Move top disk of C to B
        }
        moves++
        Break loop if A and C are both empty
    }
}
Do this loop until finished if the total number of disks is even
Else { // num_of_disks is even
    while(true) {
        Make legal move between A and C, add 1 to moves, and check if done
        If ((A < C and A is not empty) OR if C is empty) { // A and C
            Move top disk of A to C
        }
        Else {
            Move top disk of C to A
        }
        moves++
        Break loop if A and C are both empty
```

```
            Make legal move between A and B, add 1 to moves, and check if done
            If ((A < B and A is not empty) OR if B is empty) { // A and B
                Move top disk of A to B
            }
            Else {
                Move top disk of B to A
            }
            moves++
            Break loop if A and C are both empty

            Make legal move between B and C, add 1 to moves, and check if done
            If ((B < C and B is not empty) OR if C is empty) { // B and C
                Move top disk of B to C
            }
            Else {
                Move top disk of C to B
            }
            moves++
            Break loop if A and C are both empty
        }

    }
    Once finished, free the stacks and return the number of moves
    Free stacks:
    Return moves;
}

Recursive solution
recursive(disks, start, target, spare) {
    Base case
    If (disks == 1) {
        print("move disk _ from peg _ to peg _", disks, start, end)
        Moves++
        Return 0;
    }
    recursive(disks -1, start, spare, target);
    print("move disk _ from peg _ to peg _", disks, start, end)
    moves++
    recursive(disks-1, spare, target, start);
    Return moves;
}
```

stack.c

```
includes

Creates stacks
Stack *stack_create(int capacity, char name) {
    Create stack using malloc
}
Deletes stack from memory
void stack_delete(Stack *s) {
    Delete (free) the stack
}
Pops top item off of stack
int stack_pop(Stack *s) {
    Return top value of stack and move stack top down 1
}
Moves item to top of stack
void stack_push(Stack *s, int item) {
    Add item to top of stack and make that the new stack top
}
Determines if stack is empty or not
bool stack_empty(Stack *s) {
    If stack is empty return true
}
Shows the top item of stack
int stack_peek(Stack *s) {
    Return top value of stack
}
```