**1. Converting Recursive to Non-Recursive: Towers of Hanoi**

#include <stdio.h>

void hanoi(int n, char source, char auxiliary, char destination) {

if (n == 1) {

printf("Move disk 1 from %c to %c\n", source, destination);

} else {

hanoi(n - 1, source, destination, auxiliary);

printf("Move disk %d from %c to %c\n", n, source, destination);

hanoi(n - 1, auxiliary, source, destination);

}

}

int main() {

int n = 3;

hanoi(n, 'A', 'B', 'C');

return 0;

}

**Non-Recursive Version in C (using Stack)**

To convert the recursive approach to a non-recursive one, we simulate the recursive calls using a manual stack.

#include <stdio.h>

#include <stdlib.h>

typedef struct {

int n;

char source;

char auxiliary;

char destination;

int state; // 0: initial, 1: after first call

} Frame;

#define MAX 100

typedef struct {

Frame items[MAX];

int top;

} Stack;

void push(Stack\* s, Frame f) {

s->items[++(s->top)] = f;

}

Frame pop(Stack\* s) {

return s->items[(s->top)--];

}

int isEmpty(Stack\* s) {

return s->top == -1;

}

void hanoi\_iterative(int n, char source, char auxiliary, char destination) {

Stack stack;

stack.top = -1;

Frame f = {n, source, auxiliary, destination, 0};

push(&stack, f);

while (!isEmpty(&stack)) {

Frame current = pop(&stack);

if (current.n == 1) {

printf("Move disk 1 from %c to %c\n", current.source, current.destination);

} else if (current.state == 0) {

// Simulate recursive calls by pushing in reverse order

Frame f1 = {current.n - 1, current.auxiliary, current.source, current.destination, 0};

Frame f2 = {1, current.source, current.auxiliary, current.destination, 0};

Frame f3 = current;

f3.state = 1;

push(&stack, f1);

push(&stack, f2);

push(&stack, f3);

Frame f0 = {current.n - 1, current.source, current.destination, current.auxiliary, 0};

push(&stack, f0);

} else if (current.state == 1) {

printf("Move disk %d from %c to %c\n", current.n, current.source, current.destination);

}

}

}

int main() {

int n = 3;

hanoi\_iterative(n, 'A', 'B', 'C');

return 0;

}

**2. Stack Implementation in C**

**Basic Stack Using Array**

#include <stdio.h>

#include <stdlib.h>

#define MAX 100

typedef struct {

int items[MAX];

int top;

} Stack;

void init(Stack\* s) {

s->top = -1;

}

int isEmpty(Stack\* s) {

return s->top == -1;

}

int isFull(Stack\* s) {

return s->top == MAX - 1;

}

void push(Stack\* s, int item) {

if (isFull(s)) {

printf("Stack Overflow\n");

return;

}

s->items[++(s->top)] = item;

}

int pop(Stack\* s) {

if (isEmpty(s)) {

printf("Stack Underflow\n");

exit(1);

}

return s->items[(s->top)--];

}

int peek(Stack\* s) {

if (isEmpty(s)) {

printf("Stack is empty\n");

exit(1);

}

return s->items[s->top];

}