Experiments\PostfixEvaluation.c

```c
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <math.h>

#define max 100

float st[max];
int top = -1;

void push(float st[], float val);
float pop(float st[]);
float EvaluatePostfix(char exp[]);

int main() {
    float val;
    char exp[100];

    printf("\t\tEvaluating single digit postfix\n\nEnter a postfix expression: ");
    fgets(exp, sizeof(exp), stdin);
    // Remove the newline character if present
    exp[strcspn(exp, "\n")] = 0;

    val = EvaluatePostfix(exp);
    printf("\nResult: %f\n", val);

    return 0;
}

float EvaluatePostfix(char exp[]) {
    int i = 0;
    float op1, op2, value;

    while (exp[i] != '\0') {
        // If the character is a digit, push it to the stack
        if (isdigit(exp[i])) {
            push(st, (float)(exp[i] - '0'));
        }
        // Skip spaces in the expression
        else if (exp[i] == ' ') {
            i++;
            continue;
        }
        // If it's an operator, pop two operands from the stack and perform the operation
        else {
            if (top < 1) {
                printf("Error: Insufficient operands for operator '%c'\n", exp[i]);
                return -1;
            }
            op1 = pop(st);
            op2 = pop(st);

            switch (exp[i]) {
                case '+':
                    value = op2 + op1;
                    break;
                case '-':
                    value = op2 - op1;
                    break;
                case '*':
                    value = op2 * op1;
                    break;
                case '/':
                    if (op1 != 0) {
                        value = op2 / op1;
                    } else {
                        printf("Error: Division by zero\n");
                        return -1;
                    }
                    break;
                case '%':
                    value = fmod(op2, op1);
                    break;
                default:
                    printf("Unexpected character: %c\n", exp[i]);
                    return -1;
            }
            push(st, value);
        }
        i++;
    }
    // Final result should be the only value left in the stack
    if (top != 0) {
        printf("Error: The postfix expression is invalid\n");
        return -1;
    }
    return pop(st);
}

void push(float st[], float val) {
    if (top == max - 1) {
        printf("\nStack Overflow\n");
    } else {
        top++;
        st[top] = val;
    }
}

```

```c
float pop(float st[]) {
    if (top == -1) {
        printf("\nStack Underflow\n");
        return -1;
    } else {
        float val = st[top];
        top--;
        return val;
    }
}


/*
OUTPUT:-

                Evaluating single digit postfix

Enter a postfix expression: 231*+9-

Result: -4.000000

*/
```