POSTFIX - 5

Sunday, 23 March 2025 10:21 PM

PROGRAM - 5

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h> /* Added missing header for strcspn */
#define MAX_SIZE 100

// Structure to represent a stack
typedef struct {
    int top;
    int array[MAX_SIZE];
} Stack;

// Function to create a stack
Stack* createStack() {
    Stack* stack = (Stack*)malloc(sizeof(Stack));
    if (stack == NULL) {
        printf("Memory allocation failed\n");
        exit(1);
    }
    stack->top = -1;
    return stack;
}

// Function to check if the stack is empty
int isEmpty(Stack* stack) {
    return stack->top == -1;
}

// Function to check if the stack is full
int isFull(Stack* stack) {
    return stack->top == MAX_SIZE - 1;
}
```

```
// Function to push an element onto the stack
void push(Stack* stack, int item) {
    if (isFull(stack)) {
        printf("Stack overflow\n");
        exit(1);
    }
    stack->array[++stack->top] = item;
}

// Function to pop an element from the stack
int pop(Stack* stack) {
    if (isEmpty(stack)) {
        printf("Stack underflow - Invalid expression\n");
        exit(1);
    }
    return stack->array[stack->top--];
}

// Function to evaluate postfix expression
int evaluatePostfix(char* expression) {
    Stack* stack = createStack();
    int i, operand1, operand2, num;

    for (i = 0; expression[i]; ++i) {
        char ch = expression[i];

        // Skip whitespace
        if (ch == ' ' || ch == '\t')
            continue;

        // Process multi-digit numbers
        if (isdigit(ch)) {
            num = 0;
            while (isdigit(expression[i])) {
                num = num * 10 + (expression[i] - '0');
                i++;
            }
            i--; // Adjust for the loop increment
            push(stack, num);
        }
        // Process operators
```

```
        else if (ch == '+' || ch == '-' || ch == '*' || ch == '/') {
            // Check if we have enough operands
            if (stack->top < 1) {
                printf("Invalid expression: Not enough operands for operator %c\n", ch);
                exit(1);
            }

            operand2 = pop(stack);
            operand1 = pop(stack);

            switch (ch) {
                case '+': push(stack, operand1 + operand2); break;
                case '-': push(stack, operand1 - operand2); break;
                case '*': push(stack, operand1 * operand2); break;
                case '/':
                    if (operand2 == 0) {
                        printf("Error: Division by zero\n");
                        exit(1);
                    }
                    push(stack, operand1 / operand2);
                    break;
            }
        }
        else {
            printf("Invalid character in expression: %c\n", ch);
            exit(1);
        }
    }

    // Check if we have exactly one value in the stack
    if (stack->top != 0) {
        printf("Invalid expression: Too many operands\n");
        exit(1);
    }

    int result = pop(stack);
    free(stack);
    return result;
}

// Main function
```

```
int main() {
    char expression[MAX_SIZE];
    printf("Enter a valid postfix expression: ");
    fgets(expression, MAX_SIZE, stdin);

    // Remove newline character from input
    expression[strcspn(expression, "\n")] = '\0';

    int result = evaluatePostfix(expression);
    printf("Result: %d\n", result);

    return 0;
}
```