|  |  |  |
|---|---|---|
| **S.No: 1** | Exp. Name: ***Project Module*** | **Date: 2024-06-11** |

**Aim:**
Project Module
**Source Code:**

```
hello.c
```

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

#define MAX 100

typedef struct {
    int top;
    double items[MAX];
} Stack;

void push(Stack *s, double value) {
    if (s->top < MAX - 1) {
        s->items[++(s->top)] = value;
    } else {
        printf("Stack overflow\n");
    }
}

double pop(Stack *s) {
    if (s->top >= 0) {
        return s->items[(s->top)--];
    } else {
        printf("Stack underflow\n");
        return 0;
    }
}

int isOperator(char c) {
    return (c == '+' || c == '-' || c == '*' || c == '/');
}

double performOperation(double a, double b, char op) {
    switch (op) {
        case '+': return a + b;
        case '-': return a - b;
        case '*': return a * b;
        case '/': return a / b;
        default: return 0;
    }
}

int isValidExpression(const char *expr) {
    for (int i = 0; i < strlen(expr); i++) {
        if (!isdigit(expr[i]) && !isOperator(expr[i]) && !isspace(expr[i]) &&
expr[i] != '(' && expr[i] != ')') {
            return 0;
        }
    }
    return 1;
}

double evaluateExpression(const char *expr) {
    Stack values, ops;
    int i = 0, j = 0;

    while (expr[i] != '\0') {
        if (isspace(expr[i])) {
            i++;
            continue;
        }

        if (isdigit(expr[i]) || expr[i] == '.') {
            while (isdigit(expr[i]) || expr[i] == '.') {
                token[j++] = expr[i++];
            }
            token[j] = '\0';
            push(&values, atof(token));
            j = 0;
        } else if (expr[i] == '(') {
            push(&ops, expr[i]);
            i++;
        } else if (expr[i] == ')') {
            while (ops.top != -1 && ops.items[ops.top] != '(') {
                double val2 = pop(&values);
                double val1 = pop(&values);
                char op = (char)pop(&ops);
                push(&values, performOperation(val1, val2, op));
            }
            pop(&ops);
            i++;
        } else if (isOperator(expr[i])) {
            while (ops.top != -1 && isOperator((char)ops.items[ops.top])) {
                double val2 = pop(&values);
                double val1 = pop(&values);
                char op = (char)pop(&ops);
                push(&values, performOperation(val1, val2, op));
            }
            push(&ops, expr[i]);
            i++;
        }
    }

    while (ops.top != -1) {
        double val2 = pop(&values);
        double val1 = pop(&values);
        char op = (char)pop(&ops);
        push(&values, performOperation(val1, val2, op));
    }

    return pop(&values);
}

int main() {
    char expr[100];

    while (1) {
        printf("Enter an expression: ");
        fgets(expr, sizeof(expr), stdin);
        if (isValidExpression(expr)) {
            double result = evaluateExpression(expr);
            printf("Result: %lf\n", result);
        } else {
            printf("Invalid expression\n");
        }
    }

    return 0;
}
```

## Execution Results - All test cases have succeeded!

**Test Case - 1**

**User Output**

```
Hello World
```