S.No: 1	Exp. Name: Project Module	Date: 2024-06-13

Aim:

Project Module

Source Code:

hello.c

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

#define MAX 100

typedef struct {
    int top;
    char items[MAX];
} CharStack;

typedef struct {
    int top;
    int items[MAX];
} IntStack;

void pushChar(CharStack *s, char c) {
    if (s->top == (MAX - 1)) {
        printf("Stack Overflow\n");
        return;
    }
    s->items[++(s->top)] = c;
}

char popChar(CharStack *s) {
    if (s->top == -1) {
        printf("Stack Underflow\n");
        exit(1);
    }
    return s->items[(s->top)--];
}

char peekChar(CharStack *s) {
    if (s->top == -1) {
        return '\0';
    }
    return s->items[s->top];
}

void pushInt(IntStack *s, int val) {
    if (s->top == (MAX - 1)) {
        printf("Stack Overflow\n");
        return;
    }
```

```
    s->items[++(s->top)] = val;
}

int popInt(IntStack *s) {
    if (s->top == -1) {
        printf("Stack Underflow\n");
        exit(1);
    }
    return s->items[(s->top)--];
}

int precedence(char op) {
    switch (op) {
        case '+':
        case '-':
            return 1;
        case '*':
        case '/':
            return 2;
        case '^':
            return 3;
        default:
            return 0;
    }
}

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

void reverseString(char *exp) {
    int length = strlen(exp);
    for (int i = 0; i < length / 2; i++) {
        char temp = exp[i];
        exp[i] = exp[length - i - 1];
        exp[length - i - 1] = temp;
    }
}

void replaceParentheses(char *exp) {
    for (int i = 0; exp[i] != '\0'; i++) {
        if (exp[i] == '(') {
            exp[i] = ')';
        } else if (exp[i] == ')') {
            exp[i] = '(';
```

```
        }
    }
}

void infixToPostfix(char infix[], char postfix[]) {
    CharStack s;
    s.top = -1;
    int i = 0, j = 0;
    char token, temp;

    while ((token = infix[i++]) != '\0') {
        if (isdigit(token) || isalpha(token)) {
            postfix[j++] = token;
        } else if (token == '(') {
            pushChar(&s, token);
        } else if (token == ')') {
            while ((temp = popChar(&s)) != '(') {
                postfix[j++] = temp;
            }
        } else if (isOperator(token)) {
            while (s.top != -1 && precedence(peekChar(&s)) >=
precedence(token)) {
                postfix[j++] = popChar(&s);
            }
            pushChar(&s, token);
        }
    }

    while (s.top != -1) {
        postfix[j++] = popChar(&s);
    }

    postfix[j] = '\0';
}

void infixToPrefix(char infix[], char prefix[]) {
    char reversed[MAX], postfix[MAX];

    strcpy(reversed, infix);
    reverseString(reversed);
    replaceParentheses(reversed);
    infixToPostfix(reversed, postfix);
    reverseString(postfix);
    strcpy(prefix, postfix);
}
```

```
int evaluatePostfix(char postfix[]) {
    IntStack s;
    s.top = -1;
    int i = 0, op1, op2, result;
    char token;

    while ((token = postfix[i++]) != '\0') {
        if (isdigit(token)) {
            pushInt(&s, token - '0');
        } else if (isOperator(token)) {
            op2 = popInt(&s);
            op1 = popInt(&s);
            switch (token) {
                case '+': result = op1 + op2; break;
                case '-': result = op1 - op2; break;
                case '*': result = op1 * op2; break;
                case '/': result = op1 / op2; break;
                case '^': result = pow(op1, op2); break;
            }
            pushInt(&s, result);
        }
    }

    return popInt(&s);
}

int main() {
    char infix[MAX], prefix[MAX], postfix[MAX];
    printf("Enter an infix expression: ");
    scanf("%s", infix);

    infixToPrefix(infix, prefix);
    printf("Prefix expression: %s\n", prefix);

    infixToPostfix(infix, postfix);
    printf("Postfix expression: %s\n", postfix);

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

    return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Hello World