```
//ICSI 333, System Fundamentals
//Spring 2022
//Sourav
//Daniel St Andrews
//001530150

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAX_CHAR 80
/*
Program takes expression from user by keyboard input consisting
of single digit numbers and any number of spaces.
Input: Single digit expressions of any length. Base to print answer in.

Solves the input expression strictly from left to right. No operator
precedence. The result of the expression is printed both in base 10
and a base specified by the user.

Output: The result of the equation in base ten as well as a base
specified by the user.
*/

int ltrSolve(char *expression);
//Pre: Takes a char pointer to the expression being solved.
//Purpose: Evaluate the expression strictly left to right.
//Post: Returns the solved for value as an int in base 10.

char* convertToBase(int base, int input, char *str);
//Pre: Takes in the user specified base, input from ltrSolve, and a char
//array that contains the answer in a specified base at return.
//Purpose: Convert result from ltrSolve to any base.
//Post: Returns the input in user specified base.

char baseDig(int remainder, int base);
//Pre: Takers in remainder from mod operation in convertToBase, as well
//user specified base.
//Purpose: Helper method for convert to base that handles remainder for string assembly.
//Post: Returns remainder as a character appropriate to specified base.
int main()
{

    //Variables
    char input[MAX_CHAR];
    char expression[MAX_CHAR];
    char str[20];
    char *p = str;

    //Ints
    int j = 0;
    int i = 0;
    int base;
    int output;

    //The expression is not assumed to have parenthesis.
    printf("Enter an expression:");

    fgets(input, MAX_CHAR, stdin);

    //Trims the spaces off of the input
    while(input[i] != '\0')
    {
        if(input[i] != ' ')
        {
            expression[j] = input[i];
            i++;
            j++;
        }
        else
        {
            i++;
        }
    }
    expression[j] = '\0';

    //~~~~~~~~~~~~~~~~~~~~~~~~~~~~TESTING FUNCTIONALITY~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

    //Test that input was read correctly.
    printf("Input was: %s\n", expression);

    printf("What Base Do You Want the Answer In?:");
    scanf("%d", &base);

    output = ltrSolve(expression);
    p = convertToBase(base, output, str);

    printf("Output is: %d\n", output);
    printf("The output in base %d is %s\n", base, str);

    return 0;

}

int ltrSolve(char *expression)
{
    char output[MAX_CHAR];
    char var1;
    char var2;
    char op = '1';
    int solved = 0;
    int len = strlen(expression) - 1;
    int j = 0;
    int i = 0;

    while(i < len)
    {

        if((isdigit(expression[i])) && (j == 0))
        {
            var1 = expression[i];
            i++;
        }
        else
        {
            op = expression[i];
            i++;

            if(j == 0)
            {
                solved = var1 - '0';
                j++;
            }

            if(isdigit(expression[i]))
            {
                var2 = expression[i];
                i++;
            }
```

```
                int temp = var2 - '0';

                switch(op)
                {
                    case '+':
                    solved = solved + temp;
                    break;
                    case '-':
                    solved = solved - temp;
                    break;
                    case '*':
                    solved = solved * temp;
                    break;
                    default:
                    solved = solved / temp;
                    break;
                }
            }

        }
        if(op == '1')
        {
            return var1 - '0';
        }

        return solved;
    }


    char* convertToBase(int base, int input, char *in)
    {
        char ch;
        char *str = in;
        int i = 0;
        int n = 0;


        if(input == 0)
        {
            str[i] = input + '0';
            i++;
            str[i] = '\0';
            return str;
        }
        if(input < 0)
        {
            n = 1;
            input = input * -1;
        }
        while(input > 0)
        {
            str[i] = baseDig((input % base), base);
            i++;
            input = input / base;
        }
        if(n)
        {
            str[i] = '-';
            i++;
        }

        int len = strlen(in);

        //Re order string into correct order
        for(int i = 0; i < len/2; i++)
        {
            char ch = str[i];
            str[i] = str[len-1-i];
            str[len-1-i] = ch;
        }

        str[i+1] = '\0';
        return str;
    }


    char baseDig(int remainder, int base)
    {
        if(base > 9)
        {
                if(remainder >= 0 && remainder <=9)
                {
                    return (remainder + '0');
                }
                else
                {
                    return ((remainder-10) + 'A');
                }
        }
        else
        {
            return (remainder + '0');
        }
    }
```