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

/*
 *ICSI333. System Fundamentals,
 *Spring 2022,
 *TA Sourav,
 *Kiran Aziz,
 *ID: 001440162
 */



/*
 *The removeSpacesInExpression function removes spaces/whitespace from the user's input expression.
 * @param input[]: Character array holding the original user's expression.
 * @param output[]: Character array holding the user's expression with whitespace removed.
 */
void removeSpacesInExpression(char input[], char output[]){
   int i, j = 0; //Integers used for indexing.
              //'i' is the index of the input array,
              //and 'j' is the index of the output array.

   //Iterates through the input expression
   //where each symbol, besides the space character,
   //is inserted into the output expression.
   for(i = 0; i<strlen(input); i++){
       if(input[i] != ' '){
           output[j] = input[i];
           j++;
       }
   }
   //Adds terminating character to the output expression since 0,
   //in decimal, is NUL as an ASCII character.
   output[j] = 0;
}

/*
 *The splitExpression function splits the expression, after all spaces have been
 *removed, into two groups: operands and operators.
 * @param expressionWithoutSpaces[]: Character array holding the user's expression with whitespace removed.
 * @param operands[]: Integer array holding all operands from the expression.
 * @param operators[]: Character array holding all operators from the expression.
 */
void splitExpression(char expressionWithoutSpaces[], int operands[], char operators[]){
   int i;      //Setting 'i' to be the index of the expression array.
   int j = 0;  //Setting 'j' to be the index of the operands array.
   int k = 0;  //Setting 'k' to be the index of the operators array.

   //Iterates through the expression array.
   //If the symbol in the expression is a digit/number, then it is added to the operands array.
   //Otherwise, if the symbol isn't a number, it must be an operator.
   //That operator is then added to the operators array.
   for(i = 0; i<strlen(expressionWithoutSpaces); i++){
       if(isdigit(expressionWithoutSpaces[i])){
           //Converting character from expression into
           //an integer by manipulating ASCII values.
           char c = expressionWithoutSpaces[i];
           operands[j] = c - '0';

           j++;
       }else if(expressionWithoutSpaces[i] == '+' ||
              expressionWithoutSpaces[i] == '-' ||
              expressionWithoutSpaces[i] == '*' ||
              expressionWithoutSpaces[i] == '/'){
           operators[k] = expressionWithoutSpaces[i];
```

```
            k++;
        }
    }
}

/*
 *The evaluateExpression function evaluates the expression to produce a final numeric result.
 * @param operands[]: Integer array holding all operands from the expression.
 * @param operators[]: Character array holding all operators from the expression.
 * @return Final result as a signed integer after evaluating the expression.
 */
signed int evaluateExpression(int operands[], char operators[]){
    signed int result = operands[0]; //Setting result to the first number in the expression.
                                     //Every other operation in the expression is based off of this number.


    //Iterates through the operators array.
    //Each operation requires the previous result and either
    //adds, subtracts, multiplies, or divides that result
    //by the proceeding number in the expression.
    for (int n = 0; n < strlen(operators); n++) {
        if(operators[n] == '+'){
            result += operands[n+1];
        }else if(operators[n] == '-'){
            result -= operands[n+1];
        }else if(operators[n] == '*'){
            result *= operands[n+1];
        }else if(operators[n] == '/'){
            result /= operands[n+1];
        }
    }

    return result; //Returns final result of expression.
}

/*
 *The reverseString function reverses a string (or a character array in C).
 * @param str[]: String, which is a character array in C, which is to be reversed.
 */
void reverseString(char str[]){
    int i, length, temp; //Setting 'i' as the index for the string.
                         //Setting 'length' as the length of the string.
                         //Setting 'temp' as the placeholder for characters in the string.


    length = strlen(str); //strlen() is used to get the length of input string.

    //Iterates through the string/character array
    //where each character's value in the string is
    //held temporarily so it can be put into a new index.
    for (i = 0; i < length/2; i++)
    {
        temp = str[i];
        str[i] = str[length - i - 1];
        str[length - i - 1] = temp;
    }
}

/*
 *The numberConversion function converts numbers greater than 9, such as 10, 11,
 *11, 12, 13, 14, and 15as the letters A, B, C, D, E, and F, respectively, as done
 *in the hexadecimal system.
 * @param result: Final result as a signed integer after evaluating the expression from Task #1.
 * @param radix: Radix, or base of the system, specified by the user.
 * @param convertedResult[]: Character array holding the resulting conversion from decimal to the
 *                          user's desired number system.
```

```c
 */
void numberConversion(signed int result, int radix, char convertedResult[]){
   int i = 0; //Setting 'i' as the index for the convertedResult array.

   //If the final result from Task #1 is a negative number, remove the negative sign and convert
   //the number as usual using the division method. Then, add the negative sign back to the result,
   //which is just appending a minus sign to the end of the convertedResult array.  To get the proper
   //answer, the convertedResult array holding all the remainders must be reversed to get the final
   //converted result.
   if(result < 0){
      result = result * (-1);

      do{
         if(result%radix == 10){
            convertedResult[i]= 'A';
         }else if(result%radix == 11){
            convertedResult[i]= 'B';
         }else if(result%radix == 12){
            convertedResult[i]= 'C';
         }else if(result%radix == 13){
            convertedResult[i]= 'D';
         }else if(result%radix == 14){
            convertedResult[i]= 'E';
         }else if(result%radix == 15){
            convertedResult[i]= 'F';
         }else{
            convertedResult[i] = (result%radix) + '0';
         }
         result = result/radix;
         i++;
      }while(result > 0);


      convertedResult[i] = '-';
      reverseString(convertedResult);
   }

   //If the final result is not negative, proceed to convert the result from Task #1
   //as normal using the division method.  Then, reverse the convertedResult array to
   //produce the desired final converted result.
   else{
      do{
         if(result%radix == 10){
            convertedResult[i]= 'A';
         }else if(result%radix == 11){
            convertedResult[i]= 'B';
         }else if(result%radix == 12){
            convertedResult[i]= 'C';
         }else if(result%radix == 13){
            convertedResult[i]= 'D';
         }else if(result%radix == 14){
            convertedResult[i]= 'E';
         }else if(result%radix == 15){
            convertedResult[i]= 'F';
         }else{
            convertedResult[i] = (result%radix) + '0';
         }
         result = result/radix;
         i++;
      }while(result > 0);

      reverseString(convertedResult);
   }
}

/*
```

```
 *The main function performs two tasks for Project 1.
 *Task #1: Strict left-to-right evaluation of an arithmetic expression, where this value is passed to Task #2
 *Task #2: Takes the result of Task #1 and shows the result of Task #1 using an arbitrary
 *         number system with the radix specified by the user.
 *
 */
int main(){
   signed int result = 0;          //Result of Task #1
   char expression[80];            //Arithmetic expression consisting of digits 0-9, and operators +, -, *, /.
   char expressionWithoutSpaces[80]; //Arithmetic expression where all whitespace is removed.
   int operands[80];               //All operands from arithmetic expression.
   char operators[80];             //All operators from arithmetic expression.

   int radix;                  //Radix, or base of the system, specified by the user.
   char convertedResult[100];  //Character array holding the resulting conversion from decimal to the
                               //user's desired number system.

   /*TASK 1*/
   //Prompts the user for an expression.
   printf("Enter an expression: ");  fflush(stdout);
   fgets(expression, 80, stdin);                     //fgets() takes in the expression as a string value, whether it has spaces or not.
   removeSpacesInExpression(expression, expressionWithoutSpaces);  //Removes all spaces from the user's input expression.
   splitExpression(expressionWithoutSpaces, operands, operators);  //Splits expression into its operands and operators.
   result = evaluateExpression(operands, operators);               //Evaluates the expression with a strict left-to-right evaluation for
a final result.
   printf("Result: %d\n", result); fflush(stdout);                //Prints out the value of the result obtained in the previous step.


   /*TASK 2*/
   //Prompts the user to specify the radix and reads it.
   printf("Enter radix (from 2 to 16): "); fflush(stdout);
   scanf("%d", &radix);
   numberConversion(result, radix, convertedResult);       //Converts the result from Task #1 into its representation in the radix
specified by the user.
   printf("Answer: %s\n", convertedResult); fflush(stdout); //Prints out the representation of the result.

   return 0;
}
```