```c
/* Solution to Problem 1 of Assignment 4-A

   Problem Definition: Write a C program called
   ROLLNO_string_to_num.c that takes as input a decimal
   number (including sign and fractional part) as a
   string, converts the string to a number (its decimal
   equivalent value), and prints the number.

   Below is one possible solution. There are other ways to do it too.

   Author: Dr. Ramya Balachandran
   Date and time of upload: February 24, 2021 5:30 pm
   If you need help understanding this solution,
   you can send an email to bramya@smail.iitm.ac.in
*/

#include <stdio.h>
#include <string.h>
#define MAX_LENGTH 22 // 20 characters for the number + 1 for the new line
                      // character + 1 for '\0'

int main()
{
    char str[MAX_LENGTH]; // String that the user will provide as input
    float number = 0; // used to store the numeric value computed from the
                      // string. We can use double too.
    int valid_input = 1; // 1 if the string has a valid character,
                         // 0 if the character is invalid
    int index = 0; // used to access each character in str
    int length = 0; // used to store length of str
    int n_frac = 0; // used to store how many digits are there after the
                    // decimal point
    int frac_10_power = 1; // used to store by what value the digit after
                           // decimal point should be divided by
    int decimal_point = 0; // variable to store if a decimal point was read
    int negate = 0; // variable to store if the number is a negative number
    int sign_present = 0; // variable to store if a sign was provided


    printf("Enter a string with a number (max %d characters): ",
      MAX_LENGTH-2); // Message asking for a valid input
    fgets(str,MAX_LENGTH-1,stdin); // get input from user

    length = strlen(str); // get current length of str

    // Remove the new line character at the end
    if (str[length-1]=='\n')
    {
        str[length-1]='\0'; // change '\n' to '\0'
        length--; // reducing length of str by 1
    }

    // Prints the string without the '\n' at the end
    printf("Entered string is %s\n",str);


```

```c
    // The below while loop allows us to go through each character
    // in the string as long as valid_input is 1
    while ((valid_input) && (index<length))
    {
        int val; // variable to store digit value if str[index] represents
                 // a digit, -1 otherwise
        switch (str[index])
        {
            case '0':
            case '1':
            case '2':
            case '3':
            case '4':
            case '5':
            case '6':
            case '7':
            case '8':
            case '9':
            // str[index] is a digit
            {
                val = str[index]-'0'; // get the numeric value
                break;
            }
            case '.': // str[index] is a decimal point
            {
                val = -1;
                if (!decimal_point) // true if this is the first time a
                                    // decimal point occurs
                    decimal_point++;
                else // this is the second decimal point. So invalid input.
                    valid_input = 0;
                break;
            }
            case '+': // str[index] is a plus sign
            {
                val = -1;
                // sign is allowed only once and only as the first character.
                // If you want to allow spaces in the string before the sign,
                // then you can change the condition accordingly
                if (sign_present || index)
                    valid_input = 0;
                else
                {
                    negate = 0;
                    sign_present++;
                }
                break;
            }
            case '-': // str[index] is a minus sign
            {
                val = -1;
                // sign is allowed only once and only as the first character.
                // If you want to allow spaces in the string before the sign,
                // then you can change the condition accordingly
                if (sign_present || index)
                    valid_input = 0;
```

```
112                 else
113                 {
114                     negate = 1; // set to 1 to indicate that we have a
115                                 // negative number
116                     sign_present++;
117                 }
118                 break;
119             }
120             default: // invalid character
121             {
122                 val = -1;
123                 valid_input = 0;
124             }
125         }
126         if ((valid_input) && (val>=0))
127         // Condition is true when str[index] is a digit.
128         // So we have to update the value stored in number.
129         {
130             if (!decimal_point) // digit before decimal point
131                 number = number*10 + val;
132             else // digit after decimal point
133             {
134                 n_frac++;
135                 frac_10_power *=10;
136                 number = number + ((float)val/frac_10_power);
137             }
138         }
139         index++; // update index to check the next character in the string
140     }
141
142     if (valid_input)
143     {
144         // Valid number in the string
145
146         if (negate && (number>0)) // handle negative number
147             number = -number;
148         // Print the number. See the use of %.*f in the below printf
149         // statement. n_frac's value is taken in place of *. This gives
150         // us an option to print the same number of digits after
151         // decimal point as it was in the input string. (It is okay
152         // if you didn't use it in Assignment 4-A, but you have to
153         // use it in Problem 3 of Assignment 4-B.
154         printf("Number in the string is %.*f\n", n_frac, number);
155
156     }
157     else // invalid input
158         printf("ERROR: Entered string does not have a valid number.\n");
159
160     return 0;
161 }
```