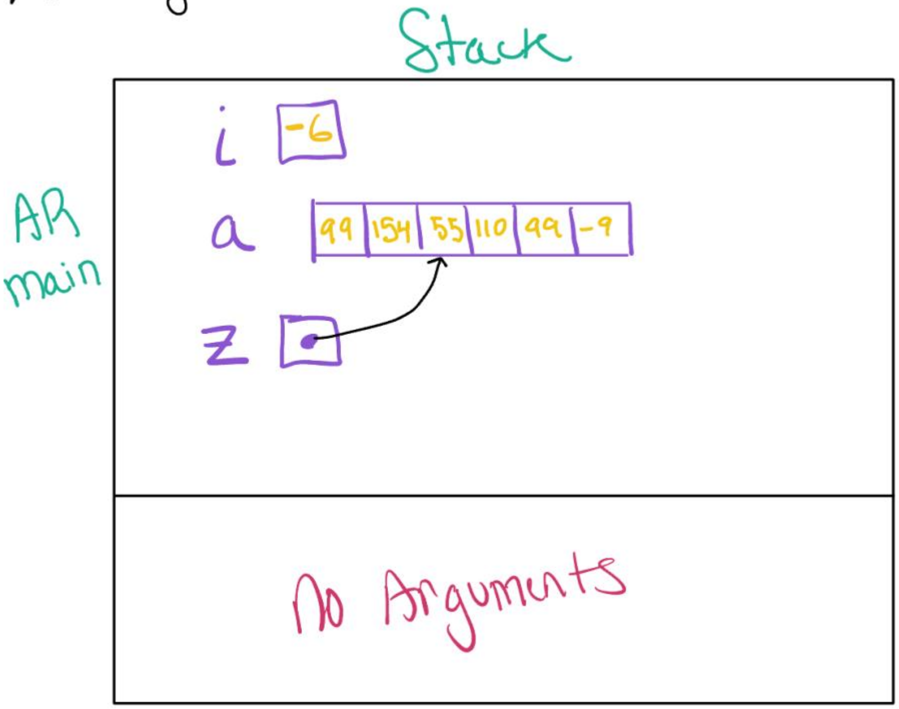
Course: Programming Fundamental - ENSF 337
Lab #: Lab 4
Instructor: Mansouri Habibabadi
Student Name: Drew Hengehold
Lab Section: B01
Date submitted: October 12, 2022
UCID: 30151823

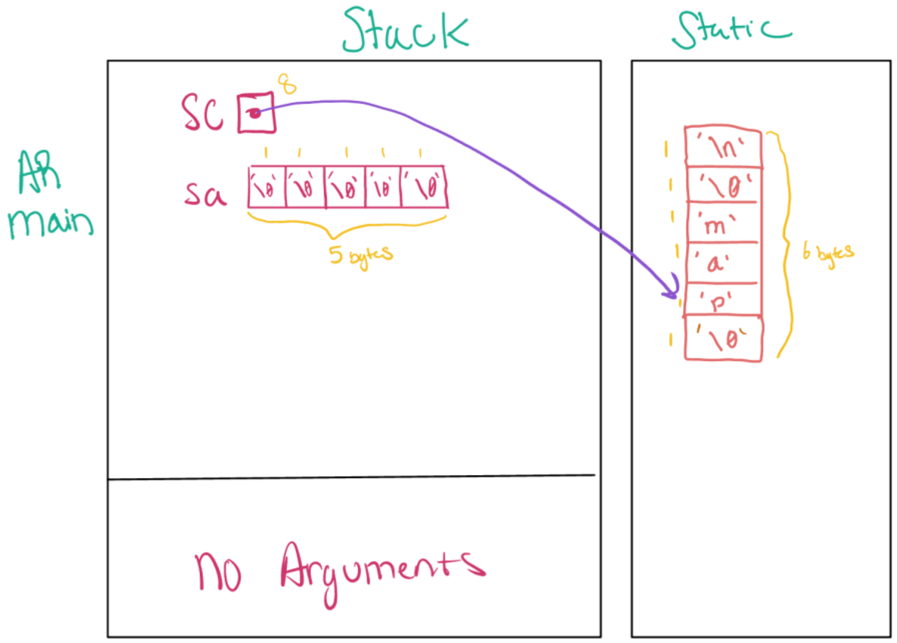
Exercise A

AR Diagram Point 1

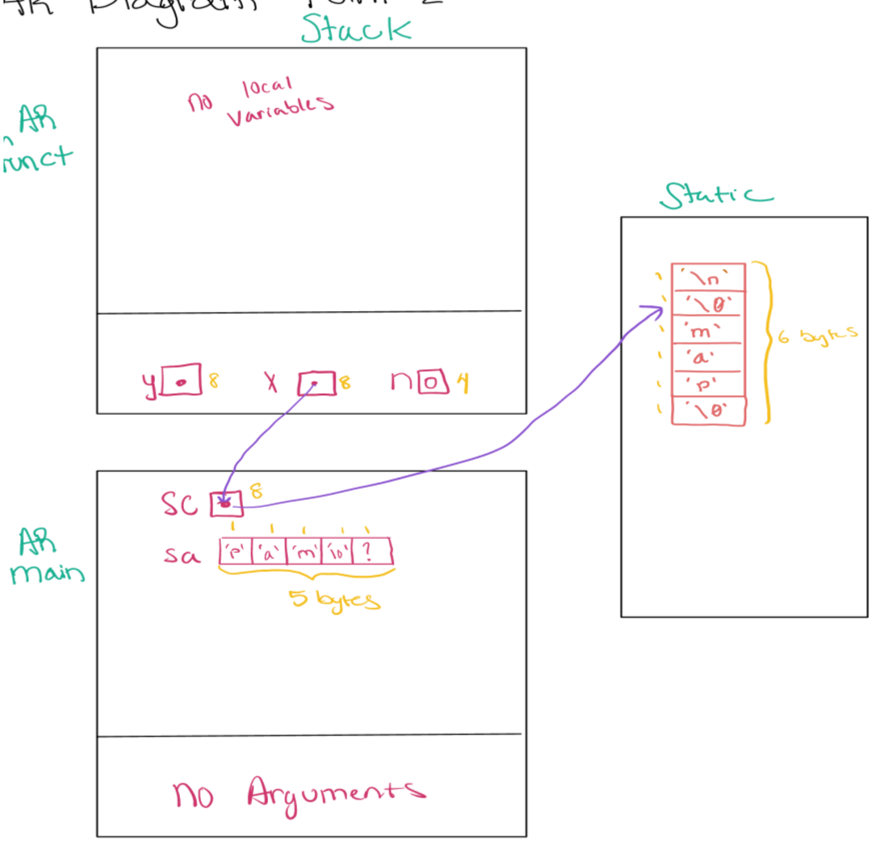


Exercise B

AR Diagram Point 1

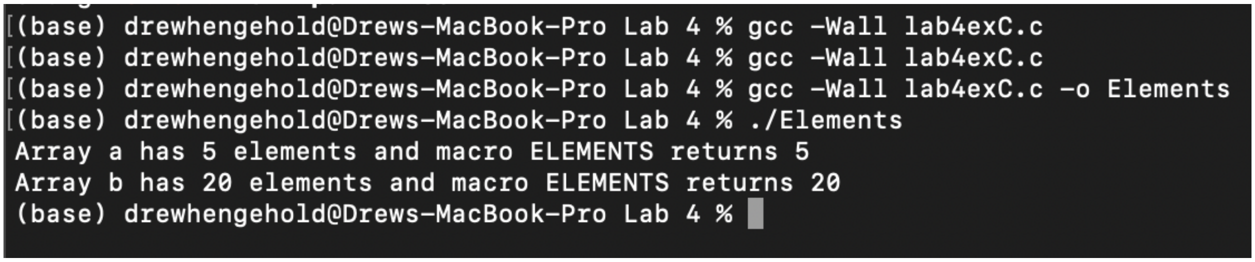


AR Diagram Point 2



Exercise C
Source Code

```
//  lab2exC.c
// ENSF 337 Lab 4 Exercise  C
//Author of Elements Drew Hengehold

#include <stdio.h>
#define ELEMENTS(x) (sizeof(x)/sizeof(x[0]))


int main()
{

    int size;
    int a[] = {45, 67, 89, 24, 54};
    double b[20] = {14.5, 61.7, 18.9, 2.4, 0.54};

    size = ELEMENTS(a);


    printf("Array a has 5 elements and macro ELEMENTS returns
%d\n", size);

    size = ELEMENTS(b);


    printf("Array b has 20 elements and macro ELEMENTS returns
%d\n", size);

    return 0;
}
```



```
[(base) drewhengehold@Drews-MacBook-Pro Lab 4 % gcc -Wall lab4exC.c
[(base) drewhengehold@Drews-MacBook-Pro Lab 4 % gcc -Wall lab4exC.c
[(base) drewhengehold@Drews-MacBook-Pro Lab 4 % gcc -Wall lab4exC.c -o Elements
[(base) drewhengehold@Drews-MacBook-Pro Lab 4 % ./Elements
 Array a has 5 elements and macro ELEMENTS returns 5
 Array b has 20 elements and macro ELEMENTS returns 20
 (base) drewhengehold@Drews-MacBook-Pro Lab 4 %
```

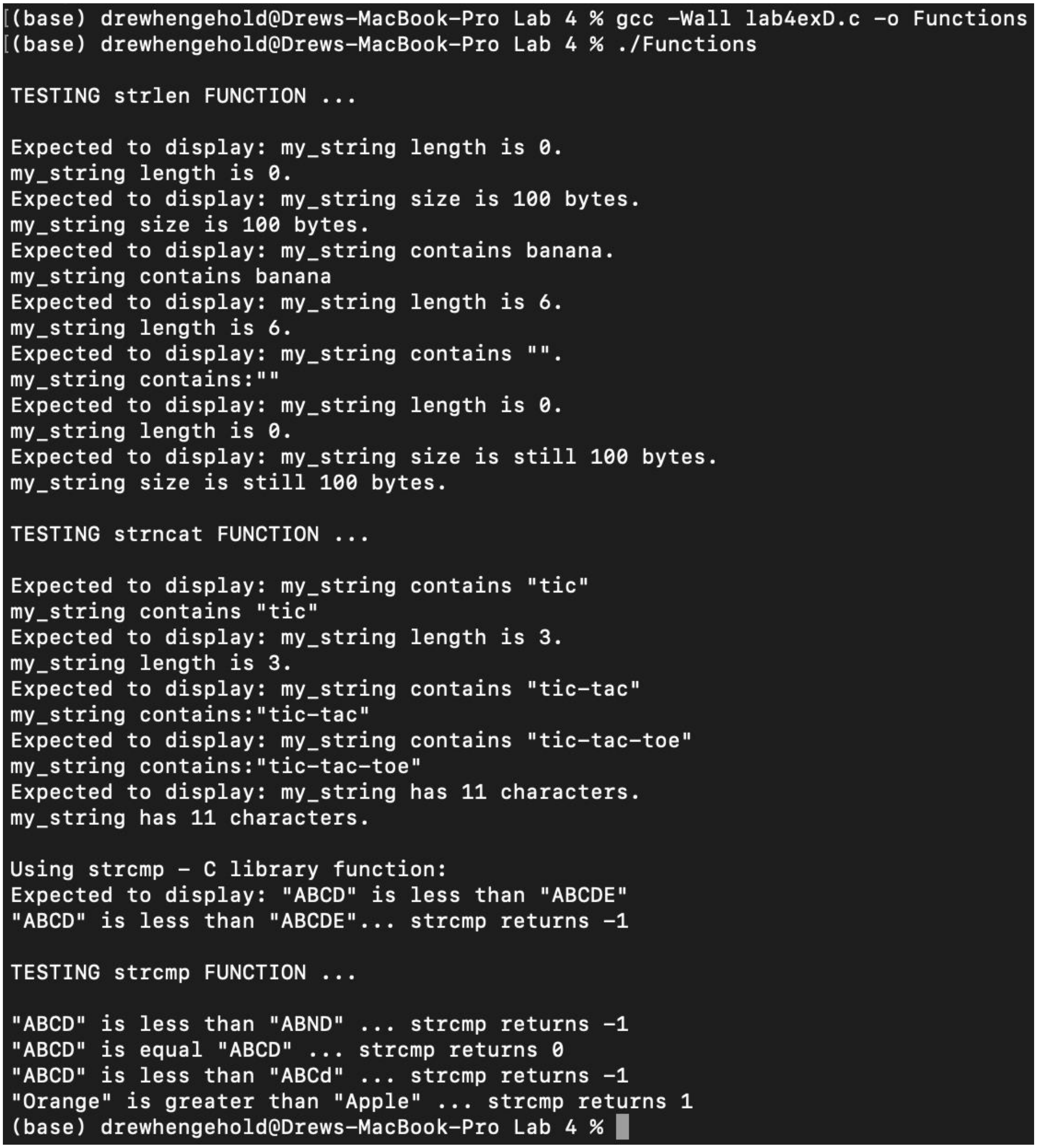
Screenshot of Output

Exercise D

Source Code

```
/*
 *  lab4exD.c
 *
 * ENSF 337 Lab 4 Exercise  D
 * AUTHOR OF FUCNTIONS Drew Hengehold
 */

#include <stdio.h>
#include <string.h>

int my_strlen(const char *s);
/*  Duplicatesmy_strlen from <string.h>, except return type is
int.
 *  REQUIRES
 *     s points to the beginning of a string.
 *  PROMISES
 *     Returns the number of chars in the string, not including
the
 *     terminating null.
 */

void my_strncat(char *dest, const char *source, int);
/*  Duplicatesmy_strncat from <string.h>, except return type is
void.
 *   dest and source point to the beginning of two strings.
 *  PROMISES
 *     appends source to the end of dest. If length of source is
more than n.
 *     Only copies the first n elements of source.
 */

int my_strncmp(const char* str1, const char* str2);
/*  Duplicatesmy_strncmp from <string.h>, except return type is
int.
 *  REQUIRES
 *     str1 points to the beginning of a string, and str2 to the
beginning of
 *     another string.
 *  PROMISES
 *     Returns 0 if str1 and str2 are idntical.
 *     Returns a negative number of str1 is less that str2.
 *     Return a psitive nubmer of str2 is less than str1.
 */

int main(void)
{
    char str1[7] = "banana";
    const char str2[] = "-tacit";
    const char* str3 = "-toe";

    char str5[] = "ticket";
    char my_string[100]="";
    int bytes;
    int length;
    int y;

    printf("\nTESTING strlen FUNCTION ... \n");

    /* using strlen function */
    length = (int) strlen(my_string);
    printf("\nExpected to display: my_string length is 0.");
    printf("\nmy_string length is %d.", length);

    /* using sizeof operator */
    bytes = sizeof (my_string);
    printf("\nExpected to display: my_string size is 100
bytes.");
    printf("\nmy_string size is %d bytes.", bytes);

    /* using strcpy C libarary function */
    strcpy(my_string, str1);
    printf("\nExpected to display: my_string contains banana.");
    printf("\nmy_string contains %s", my_string);

    length = (int) strlen(my_string);
    printf("\nExpected to display: my_string length is 6.");
    printf("\nmy_string length is %d.", length);

    my_string[0] = '\0';
    printf("\nExpected to display: my_string contains \"\".");
    printf("\nmy_string contains:\"%s\"", my_string);

    length = (int) strlen(my_string);
    printf("\nExpected to display: my_string length is 0.");
    printf("\nmy_string length is %d.", length);

    bytes = sizeof (my_string);
    printf("\nExpected to display: my_string size is still 100
bytes.");
    printf("\nmy_string size is still %d bytes.", bytes);

    printf("\n\nTESTING strncat FUNCTION ... \n");
    /* strncat append the first 3 characters of str5 to the end
of my_string */
    strncat(my_string, str5, 3);
    printf("\nExpected to display: my_string contains \"tic\"");
    printf("\nmy_string contains \"%s\"", my_string);

    length = (int) strlen(my_string);
    printf("\nExpected to display: my_string length is 3.");
    printf("\nmy_string length is %d.", length);

    strncat(my_string, str2,  4);
    printf("\nExpected to display: my_string contains \"tic-
tac\"");
    printf("\nmy_string contains:\"%s\"", my_string);

    /* strncat append ONLY up ot '\0' character from str3 -- not
6 characters */
    strncat(my_string, str3, 6);
    printf("\nExpected to display: my_string contains \"tic-tac-
toe\"");
    printf("\nmy_string contains:\"%s\"", my_string);

    length = (int) strlen(my_string);
    printf("\nExpected to display: my_string has 11
characters.");
    printf("\nmy_string has %d characters.", length);

    printf("\n\nUsing strcmp - C library function: ");
    printf("\nExpected to display: \"ABCD\" is less than
\"ABCDE\"");
    printf("\n\"ABCD\" is less than \"ABCDE\"... strcmp returns
%d", strcmp("ABCD", "ABCDE"));

    printf("\n\nTESTING strcmp FUNCTION ... \n");

    if((y = strcmp("ABCD", "ABND")) < 0)
        printf("\n\"ABCD\" is less than \"ABND\" ... strcmp
returns %d", y);

    if((y = strcmp("ABCD", "ABCD")) == 0)
        printf("\n\"ABCD\" is equal \"ABCD\" ... strcmp returns
%d", y);
```

```
    if((y = strcmp("ABCD", "ABCd")) < 0)
        printf("\n\"ABCD\" is less than \"ABCd\" ... strcmp
returns %d", y);

    if((y = strcmp("Orange", "Apple")) > 0)
        printf("\n\"Orange\" is greater than \"Apple\" ...
strcmp returns %d\n", y);

    return 0;
}

int my_strlen(const char *s){
    int counter = sizeof(s)/ *s;
    return counter;
}

void my_strncat(char *dest, const char *source, int e){
    while(*dest!= '\0'){
        dest++;

    }
    for(int i = 0; *source != '\0' && i < e; i++){
        *dest = *source;
        source++;
        dest++;
    }
    *dest = 0;
}

int my_strncmp(const char *str1, const char *str2){
    while(*str1 == '\0' && *str2 == '\0'){
        return (-1);
    }

    while(*str1 == '\0' && *str2 == '\0'){
        return 1;
    }

    while(*str1 == '\0' && *str2 == '\0'){
        if(*str1 == *str2)
            return 0;
        if(*str1 < *str2)
            return (-1);
        else
            return 1;
    }
    return 0;
}
```



```
[(base) drewhengehold@Drews-MacBook-Pro Lab 4 % gcc -Wall lab4exD.c -o Functions
[(base) drewhengehold@Drews-MacBook-Pro Lab 4 % ./Functions

TESTING strlen FUNCTION ...

Expected to display: my_string length is 0.
my_string length is 0.
Expected to display: my_string size is 100 bytes.
my_string size is 100 bytes.
Expected to display: my_string contains banana.
my_string contains banana
Expected to display: my_string length is 6.
my_string length is 6.
Expected to display: my_string contains "".
my_string contains:""
Expected to display: my_string length is 0.
my_string length is 0.
Expected to display: my_string size is still 100 bytes.
my_string size is still 100 bytes.

TESTING strncat FUNCTION ...

Expected to display: my_string contains "tic"
my_string contains "tic"
Expected to display: my_string length is 3.
my_string length is 3.
Expected to display: my_string contains "tic-tac"
my_string contains:"tic-tac"
Expected to display: my_string contains "tic-tac-toe"
my_string contains:"tic-tac-toe"
Expected to display: my_string has 11 characters.
my_string has 11 characters.

Using strcmp - C library function:
Expected to display: "ABCD" is less than "ABCDE"
"ABCD" is less than "ABCDE"... strcmp returns -1

TESTING strcmp FUNCTION ...

"ABCD" is less than "ABND" ... strcmp returns -1
"ABCD" is equal "ABCD" ... strcmp returns 0
"ABCD" is less than "ABCd" ... strcmp returns -1
"Orange" is greater than "Apple" ... strcmp returns 1
(base) drewhengehold@Drews-MacBook-Pro Lab 4 %
```

Output Example

Exercise E

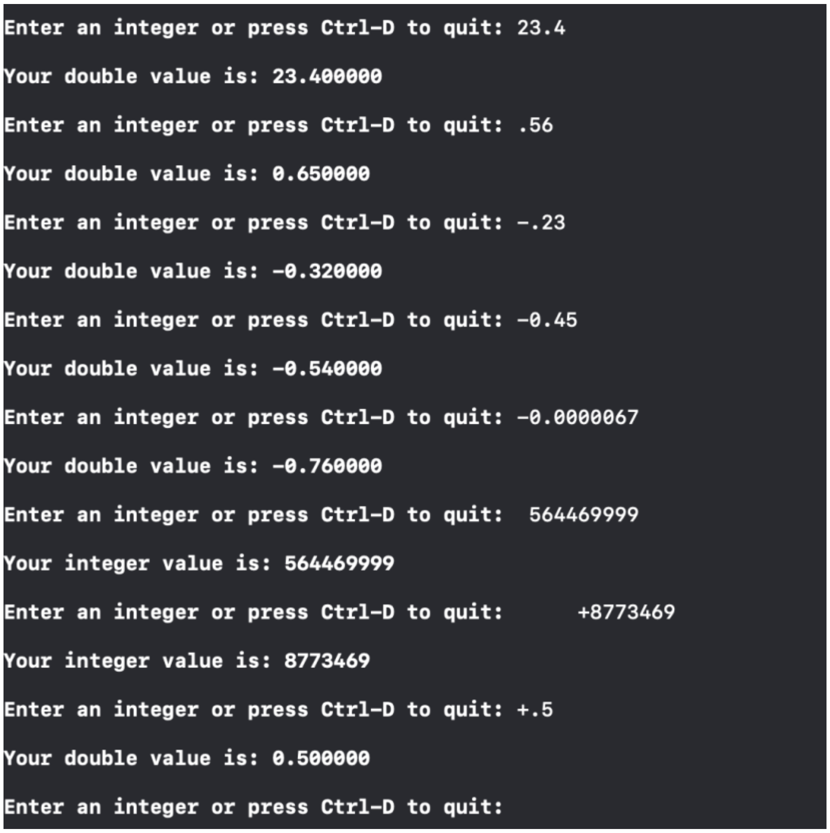
Source Code prog_two.c:

```
/* prog_two.c
 * ENSF 337 Lab 4 Exercise  E
 *
 */

#include <stdio.h>
#include <limits.h>
#include <math.h>
#include "read_input.h"

#define SIZE 50

int main(void)
{
  double n = 0;
  char digits[SIZE];

  int y = EOF;

  while (1)
    {
      printf("\n\nEnter an integer or press Ctrl-D to quit: ");
      y = read_real(digits, SIZE, &n);

      if(y == 1)
          if(fmod(n,1) == 0){printf("\nYour integer value is:
%d", (int)n);}
          else{printf("\nYour double value is: %lf", n);}
      else if(y == EOF){
     printf("\nGood Bye.\n");
     break;
      }
      else
     printf("\n%s is an invalid integer.", digits);
    }

  return 0;
}
```

Source Code read_double.c:

```
//
//  read_double.c
//  Lab4 Playground
//
//  Created by Drew Hengehold on 10/11/22.
```

```
//
#include "read_input.h"
#include <stdio.h>

int read_real(char* digits, int n, double * num){

    if(get_string(digits, n)== EOF)
      return EOF;

    if(is_valid_double(digits)){
        if(digits[0] == '-')
            *num = -convert_to_double(digits + 1);
        else if(digits[0] == '+')
            *num = convert_to_double(digits + 1);
        else
            *num = convert_to_double(digits);
        return 1;
    }


    return 0;
}
/* REQUIRES
 *   n > 0, n is large enough to handle longest expected line of
input.
 *   Array elements digits[0], ...,  digits[n-1] exist.
 *   num points to a variable.
 *
 * PROMISES:
 *   A line of user input is copied into the array, possibly
after some
 *   editing.
 *   If a double is successfully read from the input, that
double is copied
 *   into *num and 1 is returned.
 *   EOF is returned if user enters end-of-file (Ctrl-D on a
Unix-like
 *   system, and Ctrl-Z on Windows).
 *   0 is returned if the user's input can't be read as a
double.
 */


int is_valid_double(const char* digits){
    int valid = 1;
    int i;
    /* i = index where first digit should be */
    if(digits[0] == '+' || digits[0] == '-')
      i = 1;
    else
      i = 0;
//    printf("\n Valid1: %d\n", valid);

    /* Must have at least one digit, and no non-digits. */
    if (digits[i] == '\0')
      valid = 0;
    else
    //    printf("\n Valid2: %d\n", valid);
  //  printf("\n%s\n", digits);
        while (valid && (digits[i] != '\0')) {
            if((digits[i] < '0' ||  digits[i] > '9') &&
digits[i] != '.')
                valid = 0;
          //  printf("\n Valid3: %d\n", valid);
            i++;
        }
    return valid;

}
/* REQUIRES: digits points to the beginning of a valid C string.
 *
 * PROMISES: Returns 1 if all characters in digits are
 *           acceptable as characters in a real number.
 *           Otherwise returns zero.
 */


double convert_to_double(const char *digits){
    double sum = 0.0;
    double sum2 = 0.0;
    int i = 0;
    int checker =0;
    while(digits[i] != '\0') {
        if(digits[i] != '.'){
            if(checker == 1)
            {
                sum2 = sum2/10 + (((double)(digits[i] -
'0'))/10);
                i++;
            }
            else
            {
                sum = 10 * sum + (digits[i] - '0');
                i++;
            }
        }
        else
        {
            checker = 1;
            i++;
        }
    }
    sum += sum2;

    return sum;


}
/* REQUIRES: digits points to a valid C string
 *           and is_valid_double(digits) is true.
 * PROMISES: computes and returns the equivalent double value of
the string
 *           characters.
 */
```



```
Enter an integer or press Ctrl-D to quit: 23.4

Your double value is: 23.400000

Enter an integer or press Ctrl-D to quit: .56

Your double value is: 0.650000

Enter an integer or press Ctrl-D to quit: -.23

Your double value is: -0.320000

Enter an integer or press Ctrl-D to quit: -0.45

Your double value is: -0.540000

Enter an integer or press Ctrl-D to quit: -0.0000067

Your double value is: -0.760000

Enter an integer or press Ctrl-D to quit:  564469999

Your integer value is: 564469999

Enter an integer or press Ctrl-D to quit:      +8773469

Your integer value is: 8773469

Enter an integer or press Ctrl-D to quit: +.5

Your double value is: 0.500000

Enter an integer or press Ctrl-D to quit:
```

Output screenshot