Homework 3

Problem 1

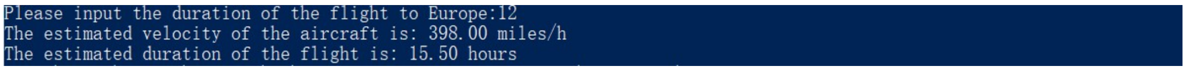
- Problem 1 is concerned with standard input and output and relevant calculations.
- Program 1 uses symbolic constants instead of magic numbers to keep the program more robust.

Program 1

```
/*----------------------------------------
Module name: program_1.c
Description: Let's continue with the travel problem. Using the library
functions printf() and scanf(),
prompt the user for the duration of the flight to Europe, in hours, as a
floating point number. Based upon the number of air miles from Seattle to
London, compute, then print out the estimated velocity of the aircraft as
a floating point number. Using the velocity you computed and an estimated
head wind of 89.6 miles per hour, compute, then print out the estimated
duration of the flight as a floating point number.
Use symbolic constants in your design rather than magic numbers as
appropriate.
Remember, the scanf() function delimits input by whitespace.
```

```
Author: Bo Yue
Rev. 0 22 Jul 2019
----------------------------------------*/

#include <stdio.h>

#define DISTANCE_SEA_LON 4781        //the distance from Seattle to London is 4781 miles
int main()
{
    int duration               = 0;
    float estimated_duration = 0.0;
    float estimated_velocity = 0.0;
    float velocity           = 0.0;

    printf("Please input the duration of the flight to Europe:");
    scanf("%d", &duration);

    estimated_velocity       = DISTANCE_SEA_LON / duration;
    printf("The estimated velocity of the aircraft is: %.2f miles/h\n", estimated_velocity);

    velocity                 = estimated_velocity - 89.6;
    estimated_duration       = DISTANCE_SEA_LON / velocity;

    printf("The estimated duration of the flight is: %.2f hours", estimated_duration);

    return 0;
}
```

Test Case



```
Please input the duration of the flight to Europe:12
The estimated velocity of the aircraft is: 398.00 miles/h
The estimated duration of the flight is: 15.50 hours
```

- Program 1 reserves two decimals for clearness and user experience.

Problem 2

-

Program 2

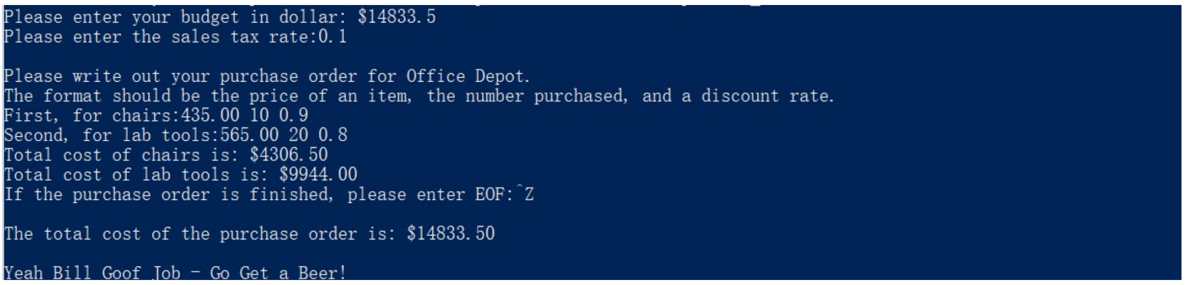
```
/*----------------------------------------
Module name: program_2.c
Description: To help Bill in engineering stores stay within his budget as he is ordering new chairs and lab tools for the EEB 137 lab, write a program to determine whether a series of purchases will come in over or under budget.
```

```
Chairs cost $435.00 each and lab stools $565.00 each, the shipping cost is
$583.00. The program starts when Bill enters a budget and sales tax rate.
As he is writing out the purchase order for Office Depot, he enters an
input line containing the price of an item, the number purchased, and a
discount rate. The program computes and prints the total cost for each
item and adds the shipping cost.
After Bill has finished putting the purchase order together, he enters the
EOF character (crtl z on a PC, ctrl d in Linux). The program then prints
the total cost of the purchase order and how it compares with the budget.
If he is under budget, the program informs him and tells him the amount of
money remaining and how many more chairs or lab stools he can purchase, if
over, it indicates so, by how much and how many chairs or lab stools he
will have to remove from his order, if spot on, it says Yeah Bill Good Job
- Go Get a Beer.
Author: Bo Yue
Rev. 0 27 Jul 2019
-----------------------------------------*/

#include <stdio.h>

#define COST_OF_CHAIRS 435.00
#define COST_OF_TOOLS 565.00
#define COST_OF_SHIPPING 583.00

// the function calculate how many chairs and tools to be removed
// form the order
void how_many_remove(float* Budget, float* Final_cost);
// the function calculate how many chairs and tools to be added
// to the order
void how_many_add(float* Budget, float* Final_cost);

int main()
{
    float budget                = 0.0;
    float* Budget               = &budget;
    float sales_tax_rate        = 0.0;

    // variables for chair-ordering
    float cost_chairs           = 0.0;
    int num_of_chairs           = 0;
    float discount_of_chairs    = 0.0;
    float total_cost_of_chairs  = 0.0;

    // variables for tool-ordering
    float cost_of_tools         = 0.0;
    int num_of_tools            = 0;
    float discount_of_tools     = 0.0;
    float total_cost_of_tools   = 0.0;

    float final_cost            = 0.0;
    float *Final_cost           = &final_cost;
    char finish                 = '0';// variable to receive EOF

    printf("Please enter your budget in dollar: $");
    scanf("%f", &budget);
    printf("Please enter the sales tax rate:");
    scanf("%f", &sales_tax_rate);

    printf("\nPlease write out your purchase order for Office Depot.\n");
    printf("The format should be the price of an item, the number
purchased, and a discount rate.\n");
    printf("First, for chairs:");
    scanf("%f %d %f", &cost_chairs, &num_of_chairs, &discount_of_chairs);
    printf("Second, for lab tools:");
    scanf("%f %d %f", &cost_of_tools, &num_of_tools, &discount_of_tools);

    cost_chairs                 = COST_OF_CHAIRS;
    cost_of_tools               = COST_OF_TOOLS;

    total_cost_of_chairs        = cost_chairs * num_of_chairs *
discount_of_chairs;
    printf("Total cost of chairs is: $%.2f\n", total_cost_of_chairs);
    total_cost_of_tools         = cost_of_tools * num_of_tools *
discount_of_tools;
    printf("Total cost of lab tools is: $%.2f\n", total_cost_of_tools);

    final_cost                  = total_cost_of_chairs +
total_cost_of_tools + COST_OF_SHIPPING;

    printf("If the purchase order is finished, please enter EOF:");
    getchar();
    // the return value for a getchar() function is an integer
    // if succeed, the return value is the ASCII value of the given
character(positive)
    // if not, the return value is EOF, namely -1(negative)
    finish = getchar();

    if (EOF == finish)
    {
        printf("\nThe total cost of the purchase order is: $%.2f\n\n",
final_cost);
    }

    if (final_cost == budget)
    {
        printf("Yeah Bill Goof Job - Go Get a Beer!\n");
    }
    else if (final_cost > budget)
    {
        printf("WARNING: You are over your budget!\n");
        printf("Below are selected options for you:\n");
        how_many_remove(Budget, Final_cost);
```

```c
    }
    else
    {
        printf("WARNING: You are under your budget!\n");
        printf("Below are selected options for you:\n");
        how_many_add(Budget, Final_cost);
    }
    return 0;
}

void how_many_remove(float* Budget, float* Final_cost)
{
    int chair       = 0;
    int tool        = 0;
    float remaining = 0.0f;

    chair     = (*Final_cost - *Budget) / COST_OF_CHAIRS; // max number of
chairs to be removed
    tool      = (*Final_cost - *Budget) / COST_OF_TOOLS;  // max number of
tools to be removed

    for (int i = 0; i <= tool; i++)
    {
        for (int j = 1; j <= chair; j++)
        {
            // remaining money
            remaining = (*Final_cost - *Budget) - i * COST_OF_CHAIRS - j *
COST_OF_TOOLS;
            if (remaining <= 0)
                {
                    break;
                }
            else
                {
                    if(remaining < 100.0)// option selection criteria is
within budget +- $100
                    {
                        printf("You should REMOVE %7d chairs and %7d lab
tools, with $%5.2f remaining.\n", i, j, remaining);
                    }
                }
        }
    }
    return;
}

void how_many_add(float* Budget, float* Final_cost)
{
    int chair       = 0;
    int tool        = 0;
    float remaining = 0.0f;
    chair      = (*Budget - *Final_cost) / COST_OF_CHAIRS; // max number of chairs to be added
    tool       = (*Budget - *Final_cost) / COST_OF_TOOLS;  // max number of tools to be added

    for (int i = 0; i <= tool; i++)
    {
        for (int j = 1; j <= chair; j++)
        {
            remaining = (*Budget - *Final_cost) - i * COST_OF_CHAIRS - j * COST_OF_TOOLS;// remaining money
            if (remaining <= 0)
                {
                    break;
                }
            else
                {
                    if(remaining < 100.0)// option selection criteria is within budget +- $100
                    {
                        printf("You can buy ANOTHER %7d chairs and %7d lab tools, with $%5.2f still remaining.\n", i, j, remaining);
                    }
                }
        }
    }
    return;
}
```

Test Case 1

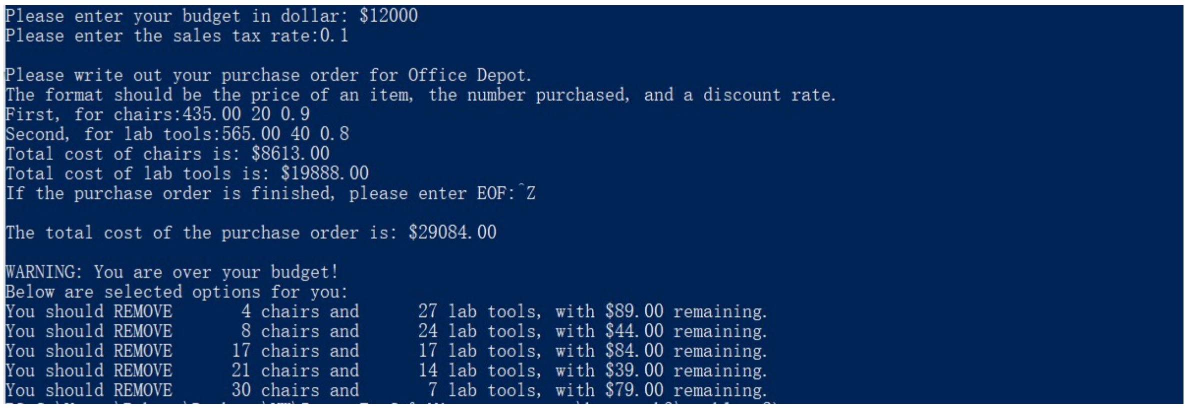
- Test Case 1 is a spot-on case.



```
Please enter your budget in dollar: $14833.5
Please enter the sales tax rate:0.1

Please write out your purchase order for Office Depot.
The format should be the price of an item, the number purchased, and a discount rate.
First, for chairs:435.00 10 0.9
Second, for lab tools:565.00 20 0.8
Total cost of chairs is: $4306.50
Total cost of lab tools is: $9944.00
If the purchase order is finished, please enter EOF:^Z

The total cost of the purchase order is: $14833.50

Yeah Bill Goof Job - Go Get a Beer!
```

Test Case 2

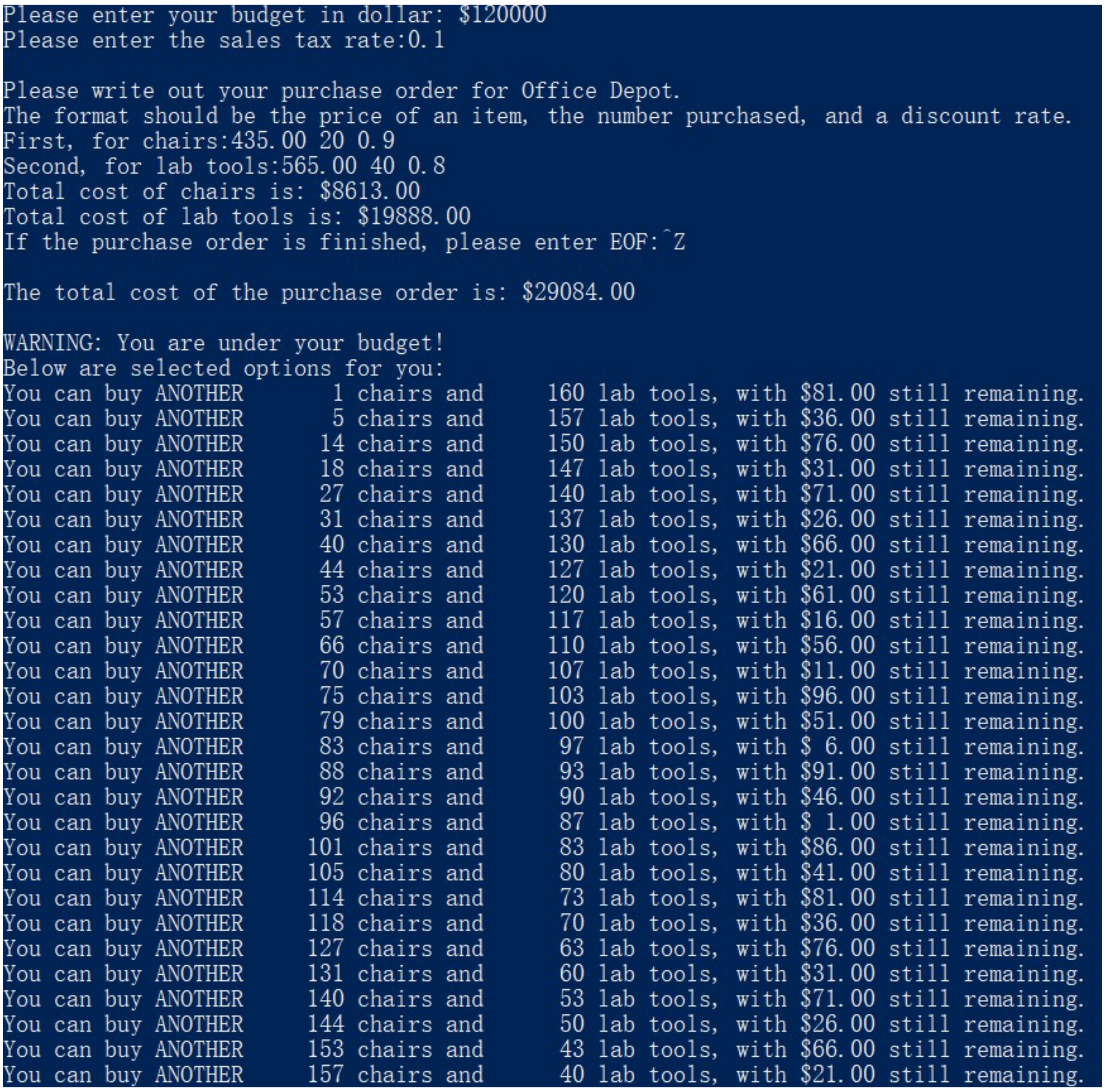
- Test Case 2 is an over-budget case.



```
Please enter your budget in dollar: $12000
Please enter the sales tax rate:0.1

Please write out your purchase order for Office Depot.
The format should be the price of an item, the number purchased, and a discount rate.
First, for chairs:435.00 20 0.9
Second, for lab tools:565.00 40 0.8
Total cost of chairs is: $8613.00
Total cost of lab tools is: $19888.00
If the purchase order is finished, please enter EOF:^Z

The total cost of the purchase order is: $29084.00

WARNING: You are over your budget!
Below are selected options for you:
You should REMOVE      4 chairs and      27 lab tools, with $89.00 remaining.
You should REMOVE      8 chairs and      24 lab tools, with $44.00 remaining.
You should REMOVE     17 chairs and      17 lab tools, with $84.00 remaining.
You should REMOVE     21 chairs and      14 lab tools, with $39.00 remaining.
You should REMOVE     30 chairs and       7 lab tools, with $79.00 remaining.
```

Test Case 3

- Test Case 3 is an under-budget case.



```
Please enter your budget in dollar: $120000
Please enter the sales tax rate:0.1

Please write out your purchase order for Office Depot.
The format should be the price of an item, the number purchased, and a discount rate.
First, for chairs:435.00 20 0.9
Second, for lab tools:565.00 40 0.8
Total cost of chairs is: $8613.00
Total cost of lab tools is: $19888.00
If the purchase order is finished, please enter EOF:^Z

The total cost of the purchase order is: $29084.00

WARNING: You are under your budget!
Below are selected options for you:
You can buy ANOTHER      1 chairs and     160 lab tools, with $81.00 still remaining.
You can buy ANOTHER      5 chairs and     157 lab tools, with $36.00 still remaining.
You can buy ANOTHER     14 chairs and     150 lab tools, with $76.00 still remaining.
You can buy ANOTHER     18 chairs and     147 lab tools, with $31.00 still remaining.
You can buy ANOTHER     27 chairs and     140 lab tools, with $71.00 still remaining.
You can buy ANOTHER     31 chairs and     137 lab tools, with $26.00 still remaining.
You can buy ANOTHER     40 chairs and     130 lab tools, with $66.00 still remaining.
You can buy ANOTHER     44 chairs and     127 lab tools, with $21.00 still remaining.
You can buy ANOTHER     53 chairs and     120 lab tools, with $61.00 still remaining.
You can buy ANOTHER     57 chairs and     117 lab tools, with $16.00 still remaining.
You can buy ANOTHER     66 chairs and     110 lab tools, with $56.00 still remaining.
You can buy ANOTHER     70 chairs and     107 lab tools, with $11.00 still remaining.
You can buy ANOTHER     75 chairs and     103 lab tools, with $96.00 still remaining.
You can buy ANOTHER     79 chairs and     100 lab tools, with $51.00 still remaining.
You can buy ANOTHER     83 chairs and      97 lab tools, with $ 6.00 still remaining.
You can buy ANOTHER     88 chairs and      93 lab tools, with $91.00 still remaining.
You can buy ANOTHER     92 chairs and      90 lab tools, with $46.00 still remaining.
You can buy ANOTHER     96 chairs and      87 lab tools, with $ 1.00 still remaining.
You can buy ANOTHER    101 chairs and      83 lab tools, with $86.00 still remaining.
You can buy ANOTHER    105 chairs and      80 lab tools, with $41.00 still remaining.
You can buy ANOTHER    114 chairs and      73 lab tools, with $81.00 still remaining.
You can buy ANOTHER    118 chairs and      70 lab tools, with $36.00 still remaining.
You can buy ANOTHER    127 chairs and      63 lab tools, with $76.00 still remaining.
You can buy ANOTHER    131 chairs and      60 lab tools, with $31.00 still remaining.
You can buy ANOTHER    140 chairs and      53 lab tools, with $71.00 still remaining.
You can buy ANOTHER    144 chairs and      50 lab tools, with $26.00 still remaining.
You can buy ANOTHER    153 chairs and      43 lab tools, with $66.00 still remaining.
You can buy ANOTHER    157 chairs and      40 lab tools, with $21.00 still remaining.
```

- For all the test cases, program 2 uses the formatting output, and always gives the user selective options to make the best use of their money. In my opinion, program 2 is very robust.

Problem 3

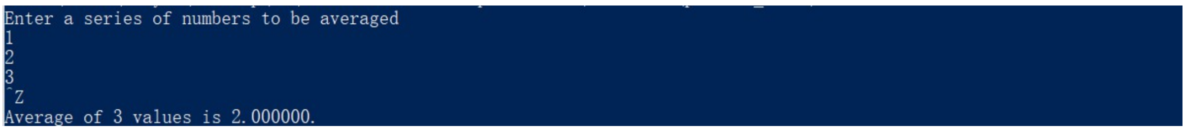
Program 3

- Problem is concerned with limits of INT datatype.

```
/*---------------------------------------
Module name: program_3.c
Description: The following program to compute the average of a collection
of values produces
incorrect results if the number of values is greater than INT_MAX, if any
input value is
greater than INT_MAX, or if the sum is greater than LONG_MAX. Rewrite the
program
to avoid these problems.
Author: Bo Yue
Rev. 0 28 Jul 2019
----------------------------------------*/

#include <stdio.h>
#include <limits.h>

int main()
{
    float next = 0;   // next input value
    long sum   = 0;   // running total
    int n      = 0;   // number of input values
    int result = 0;   // did we read another value?
    double avg = 0.0; // average of input values
    printf("Enter a series of numbers to be averaged\n");
    // read input
    // test for integer type entered
    while (1 == (result = scanf("%f", &next)))
    {
      if((next > INT_MAX) || (next < INT_MIN))                    // if
any input value is greater than INT_MAX
      {
          printf("WARNING: Your input value exceeds MAX integer!!!\nPlease
REINPUT it:");
          next = 0;
      }
      else
      {
          if(((sum + next) > INT_MAX) || ((sum + next) < INT_MIN))// if
the sum is greater than LONG_MAX
          {
              printf("WARNING: Your aggregated input value exceeds MAX
integer!!!\nPlease CONFIRM the value:");
              next = 0;
              n    = n - 1;
          }
          if(INT_MAX == n)                                        // if
the number of values is greater than INT_MAX
```

```
          {
              printf("WARNING: The number of inputs exceeds MAX
integer!!!\nNO MORE INPUT!");
              next = 0;
              n    = n - 1;
          }
          sum = sum + next; // running sum
          n   = n + 1;      // number of values entered
      }
    }
    if (result != EOF) // combination of ctrl and z keys on a PC or ctrl
and d in Linux
    {
      printf("Warning: bad input after reading %i values\n", n);
    }
    if (0 == n) // check for no numbers entered
    {
      avg = 0.0;
    }
    else // compute the average
    {
      avg = (double) sum / n;
      printf("Average of %i values is %f.\n", n, avg);
    }
    return 0;
}
```

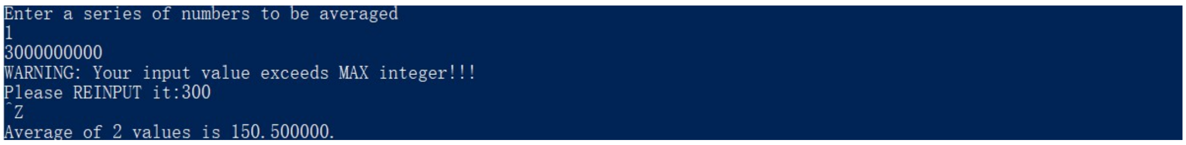
Test Case 1



```
Enter a series of numbers to be averaged
1
2
3
^Z
Average of 3 values is 2.000000.
```

- Test Case 1 is the normal input & output.

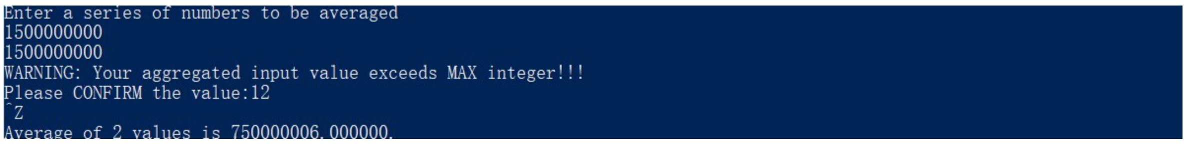
Test Case 2



```
Enter a series of numbers to be averaged
1
3000000000
WARNING: Your input value exceeds MAX integer!!!
Please REINPUT it:300
^Z
Average of 2 values is 150.500000.
```

- In Test Case 2, input value is greater than INT_MAX.

Test Case 3



```
Enter a series of numbers to be averaged
1500000000
1500000000
WARNING: Your aggregated input value exceeds MAX integer!!!
Please CONFIRM the value:12
^Z
Average of 2 values is 750000006.000000.
```

- In Test Case 3, the sum is greater than LONG_MAX
- The case "the number of values is greater than INT_MAX" might not be tested, as it is not likely to input that many numbers from my PC. However, a "for" loop can be

written to solve the testing dilemma.

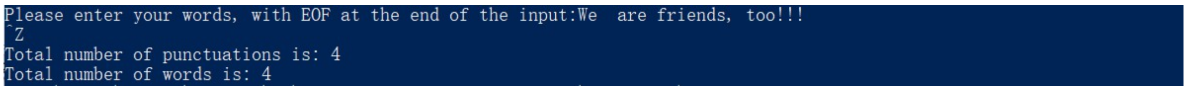
Problem 4

- Problem 4 is concerned with char input and use of *EOF*.

```
/*---------------------------------------
Module name: program_4.c
Description: Write a program that reads input data from the user, one
character at a time, until the user
enters the EOF (crtl z on a PC, ctrl d in Linux) character. As data is
being read, count the
number of words and punctuation characters that have been entered. When
the user has
finished, print out the total number of words and punctuation characters
that have been
entered.
A word is any sequence of non-whitespace characters, except punctuation
characters,
separated by whitespace characters. Look at the character testing
functions in the library
ctype.h and in Chapter 5 of the text.
Author: Bo Yue
Rev. 0 28 Jul 2019
---------------------------------------*/

#include <stdio.h>
#include <ctype.h>
#include <stdbool.h>

int main()
{
    int total_num_word  = 0;
    int total_num_punc  = 0;
    bool flag           = false;
    char receive        = '0';
    int result          = 0;

    printf("Please enter your words, with EOF at the end of the input:");
    while(1 == (result = scanf("%c", &receive)))
    {
        if(ispunct(receive))
        {
            total_num_punc = total_num_punc + 1;
        }
        if(isalpha(receive))
        {
            flag = 1;
        }
```

```c
        // when the received char turns from normal char to whitespace or
punctuation, count 1
        if(!isalpha(receive) && flag)
        {
            total_num_word = total_num_word + 1;
            flag = 0;
        }
    }

    while (result != EOF) // combination of ctrl and z keys on a PC or
ctrl and d in Linux
    {
      printf("WARNING: Bad input after reading data!!!\n");
    }

    printf("Total number of punctuations is: %d\n", total_num_punc);
    printf("Total number of words is: %d\n", total_num_word);

    return 0;
}
```

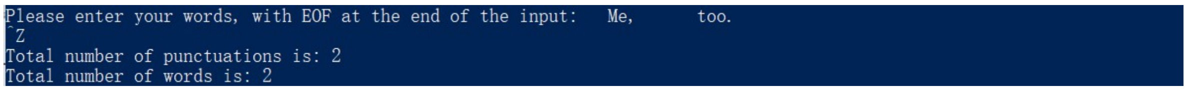
Test Case 1



```
Please enter your words, with EOF at the end of the input:We  are friends, too!!!
^Z
Total number of punctuations is: 4
Total number of words is: 4
```

- Test Case 1 is a normal sentence input.

Test Case 2



```
Please enter your words, with EOF at the end of the input:   Me,       too.
^Z
Total number of punctuations is: 2
Total number of words is: 2
```

- Test Case 2 is rather strange, with several whitespaces between words to test the robustness of the program.

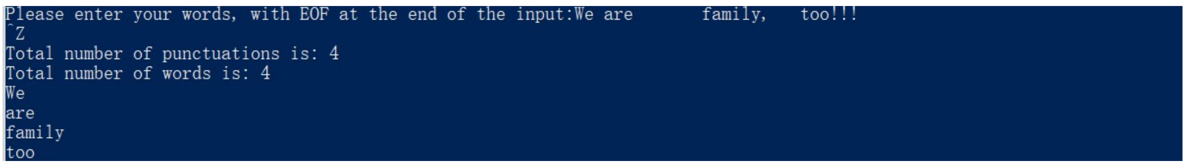
Problem 5

- Problem 5 is concerned with modifying the input and output.

```c
/*-----------------------------------------
Module name: program_5.c
Description: Modify the program in the previous problem to print each
input word on a separate line
as the data is being entered. Do not include the punctuation characters.
Author: Bo Yue
Rev. 0 28 Jul 2019
-----------------------------------------*/
```

```c
#include <stdio.h>
#include <ctype.h>
#include <stdbool.h>

#define CNT 1
int main()
{
    int total_num_word  = 0;
    int total_num_punc  = 0;
    bool flag           = false;
    char receive        = '0';
    int result          = 0;

    int last_position   = 0;
    char buf[100]       = {0};
    int cnt             = 0;

    printf("Please enter your words, with EOF at the end of the input:");
    while(1 == (result = scanf("%c", &receive)))
    {
        buf[cnt]        = receive;
        cnt++;
        if(ispunct(receive))
        {
            total_num_punc = total_num_punc + 1;
        }
        if(isalpha(receive))
        {
            flag = 1;
        }

        // when the received char turns from normal char to whitespace or
punctuation, count 1
        if(!isalpha(receive) && flag)
        {
            total_num_word = total_num_word + 1;
            flag = 0;
        }
    }

    last_position = cnt;
    while (result != EOF) // combination of ctrl and z keys on a PC or
ctrl and d in Linux
    {
      printf("WARNING: Bad input after reading data!!!\n");
    }

    printf("Total number of punctuations is: %d\n", total_num_punc);
    printf("Total number of words is: %d\n", total_num_word);

    cnt = 0;
    int m = 0;// m makes sure that only one whitespace is added between word-intervals
    for(int k = 0; k < last_position; k++)
    {
        if (isalpha(buf[cnt]))
        {
            printf("%c", buf[cnt]);
            m = 0;
        }
        else if(0 == m)
        {
            printf("\n");
            m = m + 1;
        }
        cnt++;
    }

    return 0;
}
```

Test Case



```
Please enter your words, with EOF at the end of the input:We are       family,    too!!!
^Z
Total number of punctuations is: 4
Total number of words is: 4
We
are
family
too
```

- In the Test Case, all the whitespace and punctuations are neglected as wished.

Statement

- I acknowledge that I code all the program myself.
- Signature: