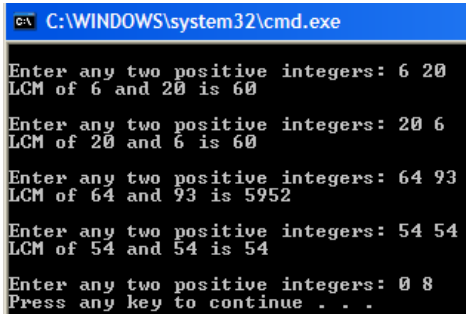
SM2-21st Computer Lab 7

Standard Algorithms, Strings and Pointers

27 Oct 2017 Friday, 6:45pm

Unzip lab7.zip. Please prepare the 5 programs in advance. You have only 2 hours in the computer lab.

Question 1:



LCM (Least Common Multiple) of two integers is the **smallest** positive multiple of both numbers. For example, LCM of 6 and 20 is 60 since 60 is the smallest positive number which is divisible by both 6 and 20. Given two positive integers S and L, where S < L. A simple algorithm to find the LCM of the two integers is to start with the larger value (L) followed by its successors. If L or its successors is divisible by both L and S, the algorithm immediately stops and the LCM is found. If S is equal to L, their LCM is S (or L). Write a full program in C language to read two positive integers from keyboard and compute their LCM. The program runs repeatedly until 0 is entered as the first integer. A session of the program execution is as shown above.

Question 2:

Temperatures are taken four times from 11:30 am to 1:30 pm daily, and stored in a text file named as **reading.inf** (unzip from Lab7.zip). Each row in the text file contains the four readings taken in one day. The text file contains the temperatures recorded in one month. *Your program should be tailored for any month.*

Contents of reading.inf for a particular month:

29.44	29.52	30.21	29.72
28.44	32.21	29.72	28.52
27.51	29.47	26.98	29.64
27.45	26.51	26.71	27.49
25.96	29.93	28.72	29.56
28.69	29.64	29.02	29.03
25.45	24.51	27.71	27.49
:	:	:	:

Name your program as **analyze.c**. You have to proceed as follows:

(i) Write a function that accepts four floating-point numbers and returns the maximum of the four numbers to its caller. The function header is as follows.

```
float findMax (float a1, float a2, float a3, float a4)
```


(ii) Write a function that accepts two pointers `start` and `end` (assume start < end) to floating-point number to delimit the memory addresses, and returns the average of all the floating-point numbers stored in the specified addresses. The function header is as follows.

```
float computeAverage (float* start, float* end)
```

(iii) Write a `main` function to process the text file, and perform statistical analysis. The function shall declare an array of floating-point numbers `max[31]`, and perform the following tasks on the recorded readings.

- Use the *findMax* function to determine the highest temperature in each day based on the four recorded readings. Store the highest temperature in the max array. Repeat this task for every row of readings.
- Use the *computeAverage* function to compute the average of the highest temperatures in the first five days of a month, and the average of the highest temperatures in the last five days of a month (call the *computeAverage* function two times from the `main` function). Display your output **on the screen** as follows.

```
        Day          Highest Temperature
       ---------------------------------
        1.                  30.21
        2.                  32.21
        3.                  29.64
        4.                  27.49
        5.                  29.93
        6.                  29.64
        7.                  27.71
        :                     :
        :                     :
        :                     :


The average of the highest temperature in the first
          five days is 29.90

The average of the highest temperature in the last
          five days is 27.29
```

Question 3:

An Identify Card (IC) number system is made up of 7 digits and an alphabet. This alphabet is calculated from the 7 digits using the modulus eleven method, and the steps are as follows:

1. Multiply each digit in the number by its weight given in the following table.

2	7	6	5	4	3	2

2. Add together the above products.
3. Divide the resulting sum by 11.
4. Subtract the remainder from 11 to give the check digit.
5. Map the check digit on the following table to obtain the alphabet.

Check Digit	1	2	3	4	5	6	7	8	9	10	11
Alphabet	A	B	C	D	E	F	G	H	I	Z	J

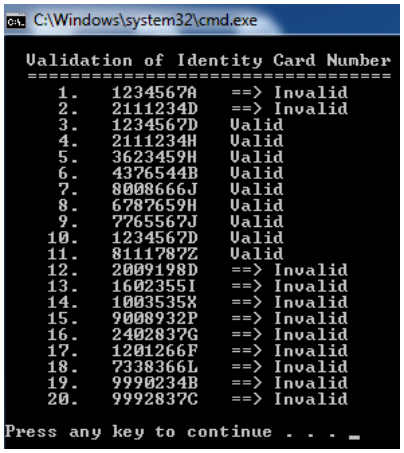
Take the string 1234567A as an example. To find out if the string is a valid IC number, we need to use the method above to calculate the alphabet as follows:

2(weight)x1(IC number) + 7(weight)x2(IC number) + 6(weight)x3 + 5x4 + 4x5 + 3x6 + 2x7 = 106.

106/11 gives a remainder of 7, and 11-7 = 4.

Using the mapping table above, the alphabet at the end of the 7 digits should be D because the check digit is 4. So, we can conclude that 1234567A is an invalid IC number.

Write a complete C program to read an input file named as **identity.inf** (unzip from Lab6.zip), check on the IC numbers and display on the screen whether the IC numbers are valid or not. The contents of **identity.inf** are shown on the right. You can assume that the input file contains 20 sets of string to be checked. The IC numbers need not be sorted. If your program runs correctly, the screen display will be as follows:




```
1234567A
2111234D
1234567D
2111234H
3623459H
4376544B
8008666J
6787659H
7765567J
1234567D
8111787Z
2009198D
1602355I
1003535X
9008932P
2402837G
1201266F
7338366L
9990234B
9992837C
```

identity.inf

Question 4:

You have learnt the **fgets** function to read a string from a file. In this lab you will use the **gets** function to read a string from the keyboard. The function declaration is as follows:

char *gets (char *str)

Parameter

str: This is the pointer to an array of chars where the C string is stored.

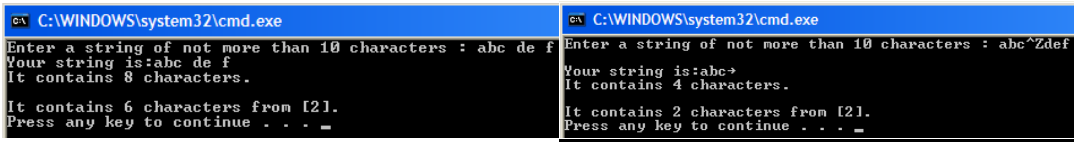
Return Value

This function returns str on success, and NULL on error or when end of file occurs while no characters have been read.

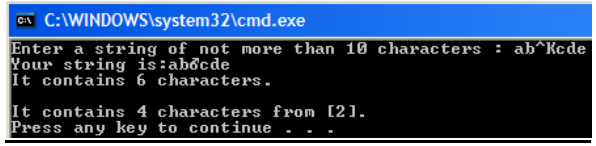
More specifically the **gets** function reads a line from stdin (usually refers to keyboard) and stores it into the string pointed to by str. It stops when either the newline character is read or when the end-of-file is reached, whichever comes first. Ctrl-z represents end-of-file from the keyboard. An example based on the **gets** function is as follows:

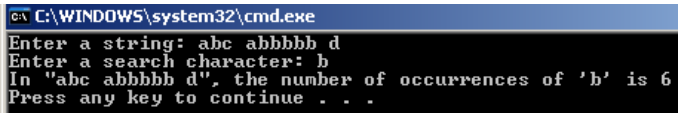

```
#include <stdio.h>          //gets.c
#include <string.h>

int main()
{
   char this1[11]; // 10+1

   printf("Enter a string of not more than 10 characters : ");
   gets(&this1[0]);

   printf("Your string is:%s", this1);

   printf ("\nIt contains %d characters.\n", strlen(&this1[0]));
   printf ("\nIt contains %d characters from [2].\n", strlen(&this1[2]));

   return 0;
}
```

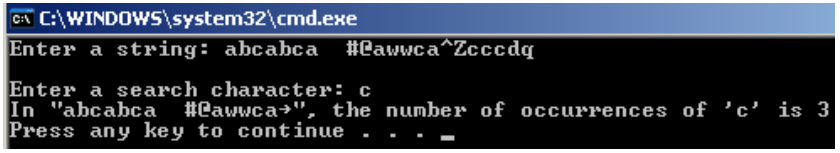
Output of gets.c



Write a full program named as count.c (unzip from Lab7.zip) to read a string of not longer than 99 characters by the **gets** function and a search character from the keyboard, and count the number of occurrences for the search character in the string. Two sessions of the program execution are as follows:

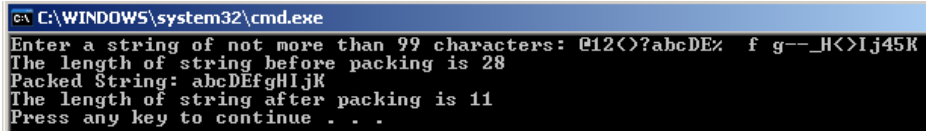






Question 5:

Write a full program named as pack.c (unzip from Lab7.zip) that contains a function with a pointer in the argument list. The program will read a string from the keyboard and remove all characters in the string except for the alphabets. The program will also pack the remaining alphabets in the same string. A session of the program execution is as follows:



Only one array of characters should be used in the program.

Use debugger whenever in doubt !!! Prepare your programs in advance.