CS 100 Final Exam – Coding – Spring 2018

Create a directory called **final** using **mkdir final** and move into that directory to complete the programs below.
Templates for programs **two.c** and **three.c** can be downloaded from Blackboard
Do NOT use the Internet while coding your programs. You can use the **cs-intro** server to test your programs if desired.
When finished, compress your **final** directory into a single (compressed) file and submit it to Blackboard.

1. Name this program **one.c** – This program assumes that you have three devices checking temperature and need to eliminate any false readings. It reads the temperatures captured by three devices (stored in three separate files), compares these values, and reports the number of potentially bad readings from each device. A value is identified as "bad" if it differs from the other two recorded values. If all three values differ, the actual temperature is considered "unknown." The exact number of data points that each file contains is given on the command line.

   As an example, consider the three files shown at the right, each having ten data values in them. The program is run as shown, generating the output shown in red. To make this example clearer to visualize, the values that differ on each line are shown in red.

Data1	Data2	Data3
72	72	72
84	83	84
56	56	56
68	68	70
81	83	83
72	72	72
68	69	69
56	56	56
68	69	70
68	68	65

```
./a.out 10 Data1 Data2 Data3
Data1 has 2 possible bad values
Data2 has 1 possible bad values
Data3 has 2 possible bad values
There are 1 unknown cases
```

2. Name this program **two.c** – This program reads two strings from standard input and determines whether or not the second string can be created from the first string by shifting the characters to the left by **N** positions (and moving the character at beginning of the string to the end). For example, the string **CDEAB** can be created from **ABCDE** by shifting the all the characters to the left by 2 positions. If it is possible to build the second string by shifting the first string to the left by **N** positions, return **N**. Note that you can build the string **ABC** by shifting **ABC** zero locations, so return zero if the two strings are identical. If it is not possible to build the second string by shifting the first string, return **–1**.

```
#include <stdio.h>
#include <string.h>
int sameWhenShifted(char *, char *);
int main(int argc, char *argv[]) {
  char s1[100], s2[100];
  printf("Enter string 1 : ");
  scanf("%s", s1);
  printf("Enter string 2 : ");
  scanf("%s", s2);
  int ans = sameWhenShifted(s1, s2);
  if ( ans != -1 )
    printf("%s shifts %d to get %s\n", s1, ans, s2);
  else
    printf("Cannot build %s from %s\n", s2, s1);
  return 0;
}
```

```
./a.out
Enter string 1 : ABCDE
Enter string 2 : DEABC
ABCDE shifts 3 to get DEABC
```

```
./a.out
Enter string 1 : XYZ
Enter string 2 : YZX
XYZ shifts 1 to get YZX
```

```
./a.out
Enter string 1 : ABC
Enter string 2 : D
Cannot build D from ABC
```

```
./a.out
Enter string 1 : ABC
Enter string 2 : BCD
Cannot build BCD from ABC
```

3. Name this program **three.c** – This program takes a linked list of words and counts certain types of words. Specifically, it counts the number of words that are either all lower-case or all upper-case letters. For example, in the list of words below, the count returned would be **2** (**alabama** and **TIDE**).

```
BigAl alabama TIDE UofA UA#1 1831 CSisFun!
```

The main program, a linked list add routine (add-at-front), and a print routine are provided for you on Blackboard. Do not modify any of this code. You simply need to write **countSame**.