# time.h another example

Read this

```
#include <stdio.h>
     #include <time.h>

        #define PST (-8)
        #define CET (1)

     int main ()
       {
                time_t raw_time;
                struct tm *ptr_ts;

                time ( &raw_time );
                ptr_ts = gmtime ( &raw_time );

                printf ("Time Los Angeles: %2d:%02d\n",
                     ptr_ts->tm_hour+PST, ptr_ts->tm_min);
                printf ("Time Amsterdam: %2d:%02d\n",
                     ptr_ts->tm_hour+CET, ptr_ts->tm_min);
         return 0;
        }
```

A) Write a program that calculates the time differences between Houston and:

- NYC
- LA
- Juneau
- London
- Moscow
- Tokyo
- Ankara
- Sydney
- Wellington

B) Copy-past this problem. Then, use this information along with IF statements to print on the screen whether it is a good idea to call people in those cities over the phone, at noon Houston time (CT).

# Exercise -- Functions

**stdDeviation.c** received over email.

It's a program that computes the standard deviation of dynamically read data. What's new?

Run the program in codechef/code Blocks.

Then:

**Create a function in this program.** The idea is that the main() function only takes care of reading data, while the new function takes care of math.

Sorting arrays by selection. You will receive “SelectionSort.c” via email. Copy paste to your C programming console (codechef, visual studio, etc.). Read it and make sure you understand what it does.

Then, substitute ALL ??? by the correct code so that it works properly.

Note the use of functions and pointers.

Can you think of a different way of doing the same task? If so, then share your thoughts!

Sorting arrays by randomizing. You will receive “RandomSort.c” via email. Copy paste to your C programming console (codechef, visual studio, etc.). Read it and make sure you understand what it does.

Then, substitute ALL ??? by the correct code so that it works properly.

Note the use of functions, random numbers, #include variables and pointers.

Can you think of a different way of doing the same task? If so, then share your thoughts!

Read this material on how strings are dealt with, and the importance of their use in C.

https://chortle.ccsu.edu/CPuzzles/PartS/stringsIntro.html

## String Puzzles — Introduction

This section is about null-terminated strings. Many of the puzzles are similar to the functions in the standard strings library of most C environments.

### 1. Null-terminated strings

A null-terminated string is a sequence of bytes in memory where each byte encodes a single character using ASCII. The last character is followed by a byte containing all zero bits. That byte is called a null byte. For example, the string "The game is afoot!" is stored as:


In this picture, each yellow square represents a byte. The slash in the last cell represents the null byte. The empty cells represent the space character, represented in ASCII as the pattern 0x20.

### 2. ASCII

Historically, C programs have encoded characters as bit patterns using the ASCII encoding scheme. Although various libraries use other encoding schemes, ASCII remains the most popular and is used in these puzzles.

The following table shows the ASCII encoding scheme. Each character is encoded as an 8-bit pattern. The table shows the pattern using the hexadecimal name for the pattern. The column heading gives the 1's place digit of the hex and the row heading gives the 16's place digit of the hex. For example, find the character 'K' in the central (yellow) portion of the chart. The row heading is 4 and the column heading is B so the bit pattern is 0x4B, or `0100 1011`.

| | 1's place | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 16's place | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| 0 | NUL | SOH | STX | ETX | EOT | ENQ | ACK | BEL | BS | HT | LF | VT | FF | CR | SO | SI |
| 1 | DLE | DC1 | DC2 | DC3 | DC4 | NAK | SYN | ETB | CAN | EM | SUB | ESC | FS | GS | RS | US |
| 2 | SP | ! | " | # | $ | % | & | ' | ( | ) | * | + | , | - | . | / |
| 3 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | : | ; | < | = | > | ? |
| 4 | @ | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O |
| 5 | P | Q | R | S | T | U | V | W | X | Y | Z | [ | \ | ] | ^ | _ |
| 6 | ` | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o |
| 7 | p | q | r | s | t | u | v | w | x | y | z | { | \| | } | ~ | DEL |