CSE2312-001 (Fall 2020)
Homework #3

Notes:

- All numbers are in base-10 unless otherwise noted.
- If part of a problem is not solvable, explain why in the answer area.
- Print out the form and handwrite your answers in the spaces below.
- Place the hw3.s file and the scanned answers to problem 1 in a single zip file with name lastname_hw3.zip, where lastname is your last name as listed in MyMav.
- Submit the single zip file to Canvas before 11:59:00pm on November 5, 2020.
- Make sure that the code follows the procedure call standards for ARM architecture (see IHI0042F section 5.1), with emphasis on this requirement: “A subroutine must preserve the contents of the registers r4-r8, r10, r11 and SP (and r9 in PCS variants that designate r9 as v6).” (in other words, push and pop R4-11 if you need to use them, as shown in the vector.s examples in class)

1. Suppose that BUSINESS6 structure is defined as:

```
typedef struct _BUSINESS5
{
    uint32_t taxId;
    char name[23];
    char direction;
    char street[31];
    uint32_t addNo;
    char city[28];
    char state[3];
    uint32_t zip;
} BUSINESS6;
```

Show the relative offset of each field in the structure from the beginning of the structure for the unpacked (default alignment) case:

Show the relative offset of each field in the structure from the beginning of the structure for the packed case:

2. Write assembly functions that implement the following C functions:

a. bool isStrSame(const char str1[], const char str2[])
   // returns 1 if the strings match, 0 otherwise

b. void strConcatenate(char strTo[], const char strFrom[])
   // adds the contents of string strFrom to strTo
   // note: strTo must have enough space to hold strFrom and strTo

c. uint32_t sumU16_32(const uint16_t x[], uint32_t count)
   // returns sum of the values in the array (x) containing count entries.

d. int32_t sumS32(const int32_t x[], uint32_t count)
   // returns sum of the values in the array (x) containing count entries.

e. uint32_t countInRange(const int32_t x[], int32_t low, int32_t high, uint32_t count)
   // returns number of values in the array (x) containing count entries that are >= low and <= high

f. void rightStringFull(char* strOut, const char* strIn, uint32_t length)
   // input: array (strIn) containing the input string, and the number of characters to extract (length)
   // output: array (strOut) containing length number of strIn characters from the end of the array or an empty string if the length is larger than can be accommodated
   // strIn = ‘abcdef’, length = 5 → returns strOut = ‘bcdef’
   // strIn = ‘abcdef’, length = 7 → returns strOut = ‘’

g. void rightStringTrunc(char* strOut, const char* strIn, uint32_t length)
   // input: array (strIn) containing the input string, and the number of characters to extract (length)
   // output: array (strOut) containing up to, but not exceeding length number of strIn characters from the end of the array
   // strIn = ‘abcdef’, length = 5 → returns strOut = ‘bcdef’
   // strIn = ‘abcdef’, length = 7 → returns strOut = ‘abcdef’

h. void sortAscending (uint32_t x[], uint32_t count)
   // input: array (x) containing count entries
   // output: array (x), with the values sorted in ascending order

i. uint16_t decimalStringToUint16(const char* str)
   // convert the null-terminated string (str) to an unsigned 16-bit integer
   // treat the string as representing a decimal number

// if a character other than 0, 1, 2, 3, 4, 5, 6, 7, 8, or 9 is present or the value is too large, return 0

j. int8_t decimalStringToInt8(const char* str)
// convert the null-terminated string (str) to a signed 8-bit integer
// treat the string as representing a decimal number
// if a character other than 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, or – is present or the value is out of range, return 0

k. uint8_t hexStringToUint8(const char* str)
// convert the null-terminated string (str) to an unsigned 8-bit integer
// treat the string as representing a hexadecimal number
// if a character other than 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, or F is present or the value is too large, return 0

l. void uint32ToBinaryString (char* str, uint32_t x)
// convert the unsigned integer (x) to a null-terminated string (str) representing a binary number

m. int32_t findCityAligned (const char city[], const BUSINESS6 business[], uint32_t count)
// returns the index of the first entry in the array (business) containing count entries which matches the requested city. If the city is not found, return a value of -1. You can assume that C default alignment is used for this problem.

n. int32_t findCityPacked (const char city[], const BUSINESS6 business[], uint32_t count)
// returns the index of the first entry in the array (business) containing count entries which matches the requested city. If the city is not found, return a value of -1. You can assume that C packing is used for this problem.

Write the solution of each of these functions in a single file hw3.s with functions being callable from a C program. You do not need to send the .c file.