Programming #1 -- simple C aliasing problem

Re-submit Assignment

Due Sep 1 by 11:59pm **Points** 30 **Submitting** a file upload **File Types** pdf
Available until Sep 3 at 11:59pm

Assignment #1 -- Simple example of type casting and pointers:

Useful link -- **ASCII** (http://www.asciitable.com/) table

Programming languages are always run on hardware. Whether a language is compiled or interpreted, the instructions that run the program are actually in machine code (INTEL, Motorola, etc). Additionally, languages must use memory to hold information. In most cases, programming languages will use a byte (or several bytes) to represent characters, and machine words to represent integers.

The underlying hardware does not distinguish between data being character data or integer data (or even other data). It is the machine instruction which makes the differentiation.

This problem helps demonstrate how the C programming language can utilize the same data location for integers and character strings through the use of pointers.

You are to create a C program which fills an integer array with integers and then you are to cast it as a "char * and print it out the string relating the "char * The output of the string should be your first and last name with proper capitalization, spacing and punctuation.

The 'C' language has a 32 bit (4 byte) data word structure. We will be using ASCII (single byte values) for characters You are to fill in the 4 bytes of each integer with characters from your name. This means you need to have 4 characters in each 'C' integer word. Please note that INTEL architectures are little Endian.

The idea, is to have your program allocate a chunk of data via and array of integers. You are then to place the correct integers into the array (4 bytes at a time), and then end the final integer with a '0', because C uses the null character to indicate end of string.

Your program should have structure similar to:

```
main()
{
    int A[100];
    char *S;

A[0]=XXXX;
A[1]=YYYY;

...

A[n]=0;  -- because C strings are terminated with NULL

...

printf("My name is %s\n",S);

}
```

Requirements:

1) Your code shall be properly commented include name , date, input, output, preconditions and postconditions

2) A screen shot of you program running

3) Answers to the following questions

a) in what memory segment is the array allocated? Give proof that your answer is correct
b) in what memory segment is the pointer to the array allocated? Give proof that your answer is correct
c) how can you make your array be in another segment? Show how you did this and show proof
d) What endianness was the computer you ran your problem on?
e) Why is there a difference between little and big endian? Which one is better? Provide a source

4) Do you we need to fill the entire last interger with '0', or can we just fill in the last byte with '0'. Show an experiment that shows this (make sure you pay attention to endianness and ensure that your other bytes are NOT 0 when doing the experiment.