**Universidad Autónoma de Guadalajara**

Ingeniería Electrónica Biomédica

System designing with Microprocessors

*“*Practice 1. Development of functions in C using cortex M0+”

Jesús Arnoldo Zerecero Núñez

2885993

Practice 1. Development of functions in C using cortex M0+

**Introduction**

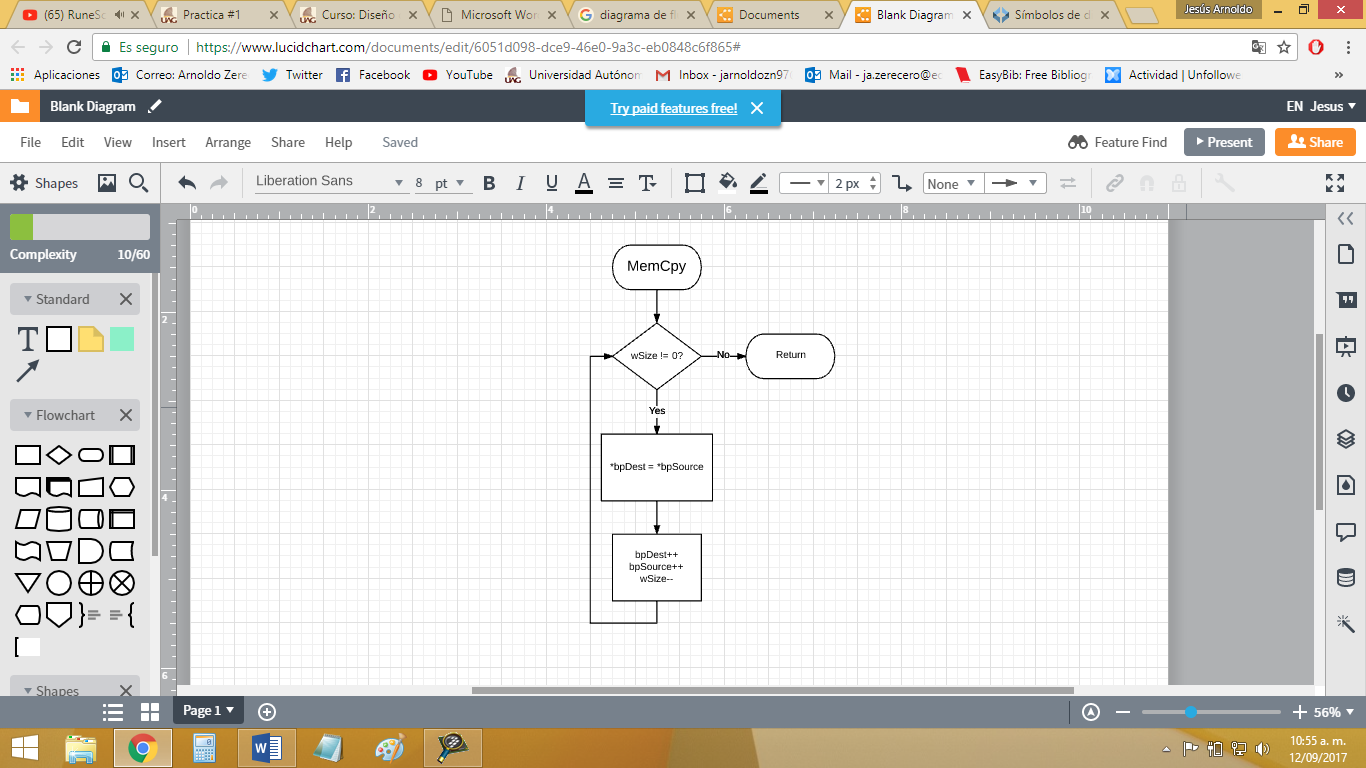
C is a general purpose programming language associated to UNIX operating system. It’s a high level language orientated to objects (characters, numbers, bits and memory spaces). It has a great portability by itself, yet it can be increased further by the inclusion of C99 data types (using the “Stdint.h” header). C language is used vastly along a great variety of software, including compilers and text editors.

A code made using C language may include a certain number of **functions**, dividing the work in each of them with the purpose of making a much cleaner and understandable code. Some functions are already defined in C headers. These are included in different files with .h extension. After including the corresponding .h file, the function can be called directly in a single line of code, by just passing the parameters the function will work with. If a function isn’t predefined, what the function does internally must also be included in the code, outside the main function.

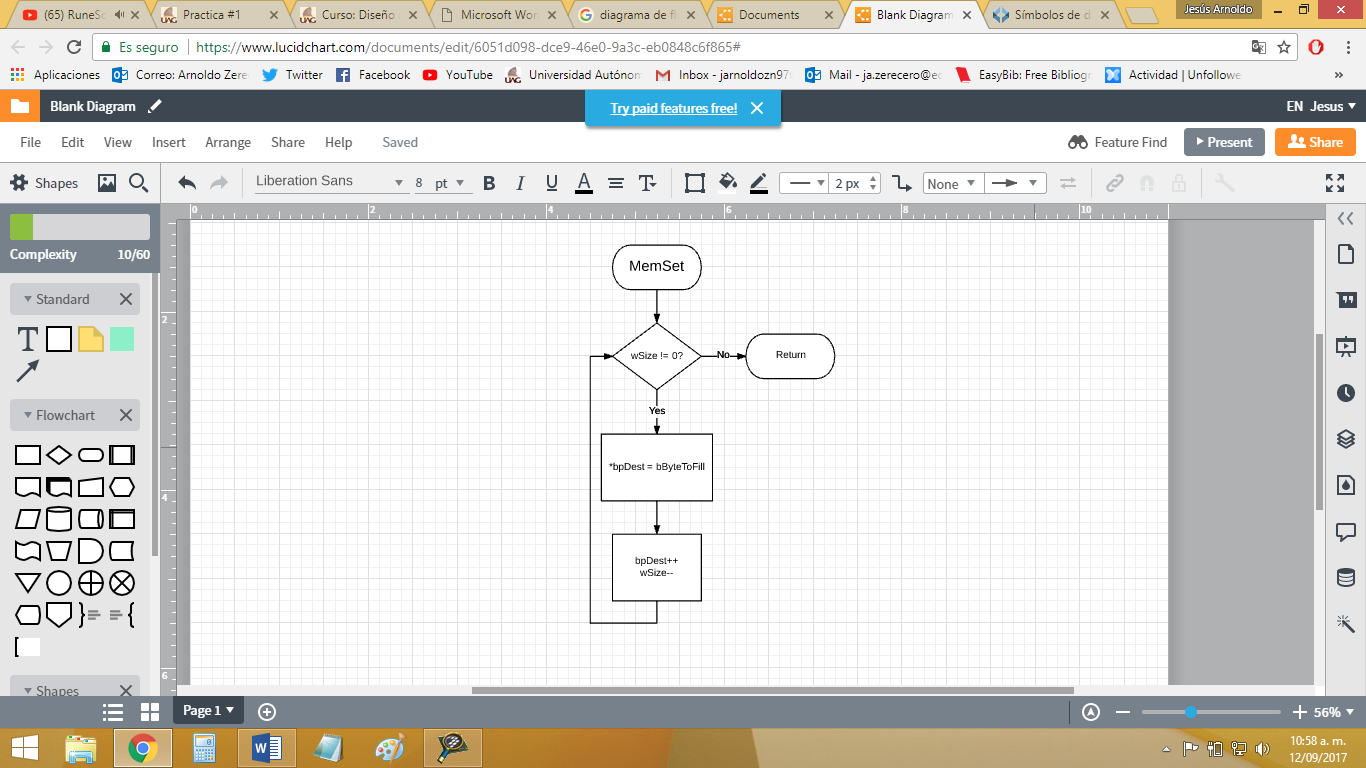
In this practice, 9 different C functions will be created. These functions already exist in C standard libraries, but no header files are allowed to be used. The functions must be coded from scratch, using our own logic.

**Development.**

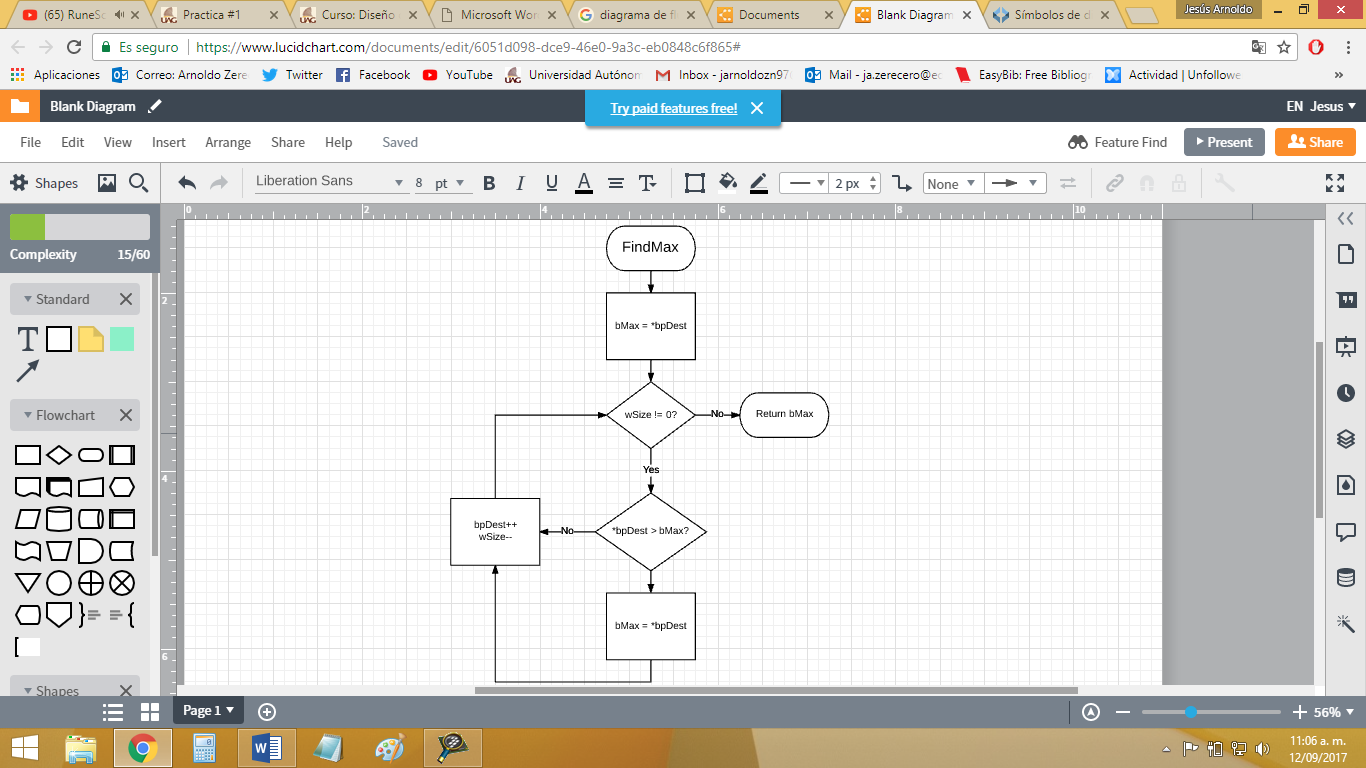
**MemCpy:**



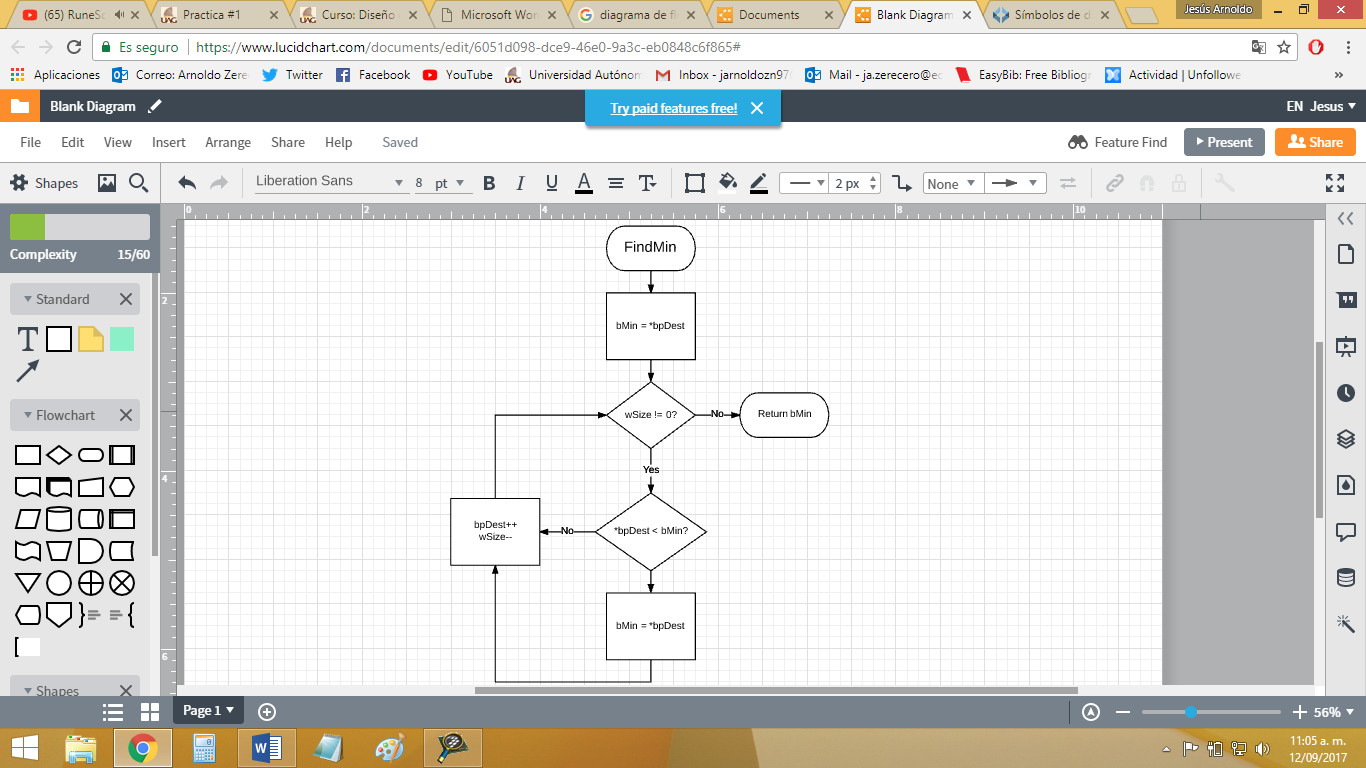
**MemSet:**



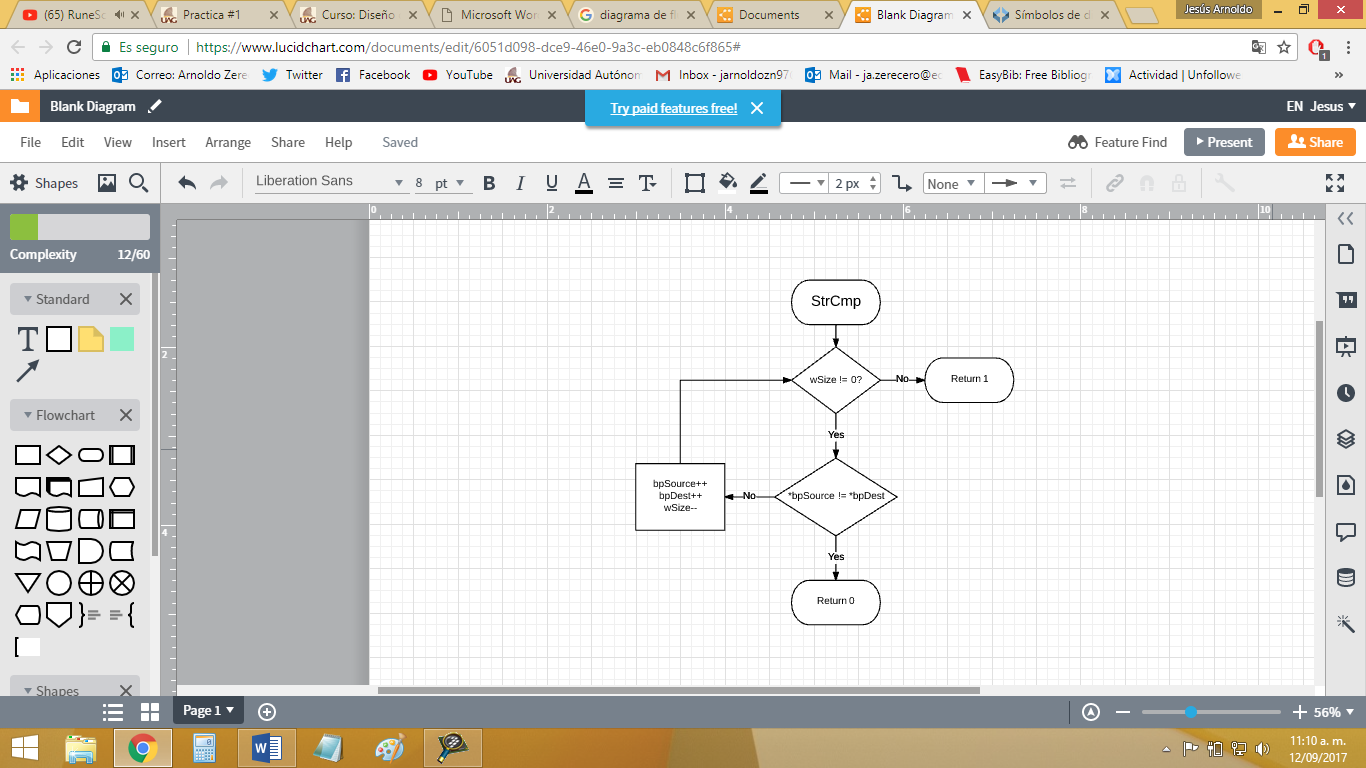
**FindMax:**



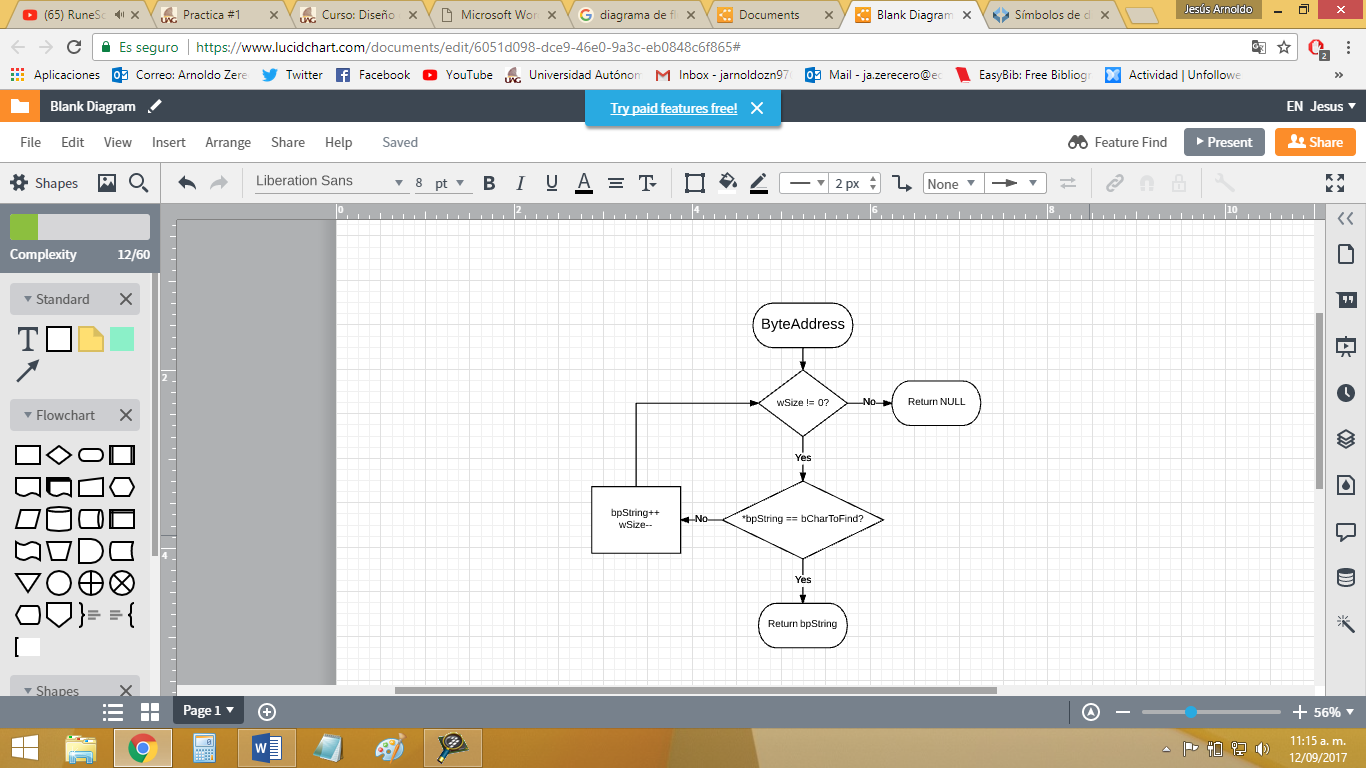
**FindMin:**



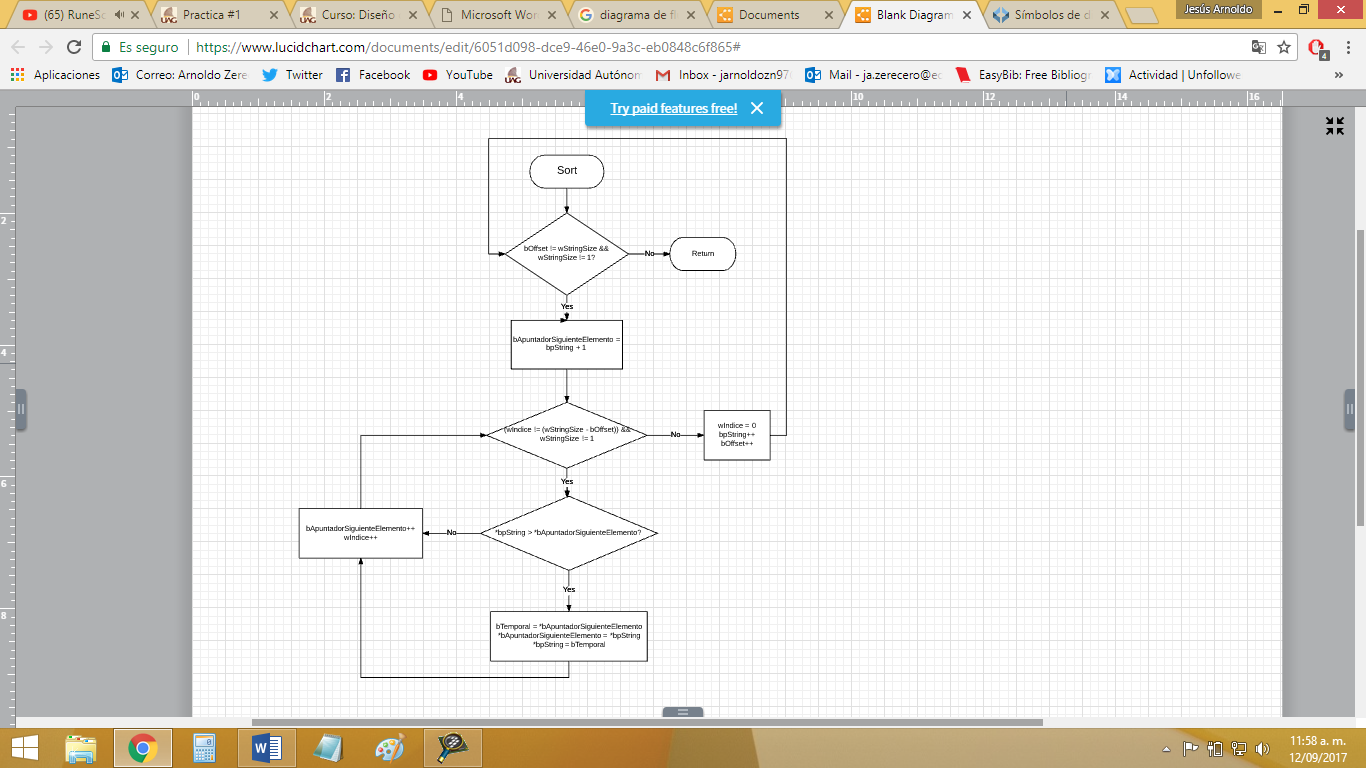
**StrCmp:**



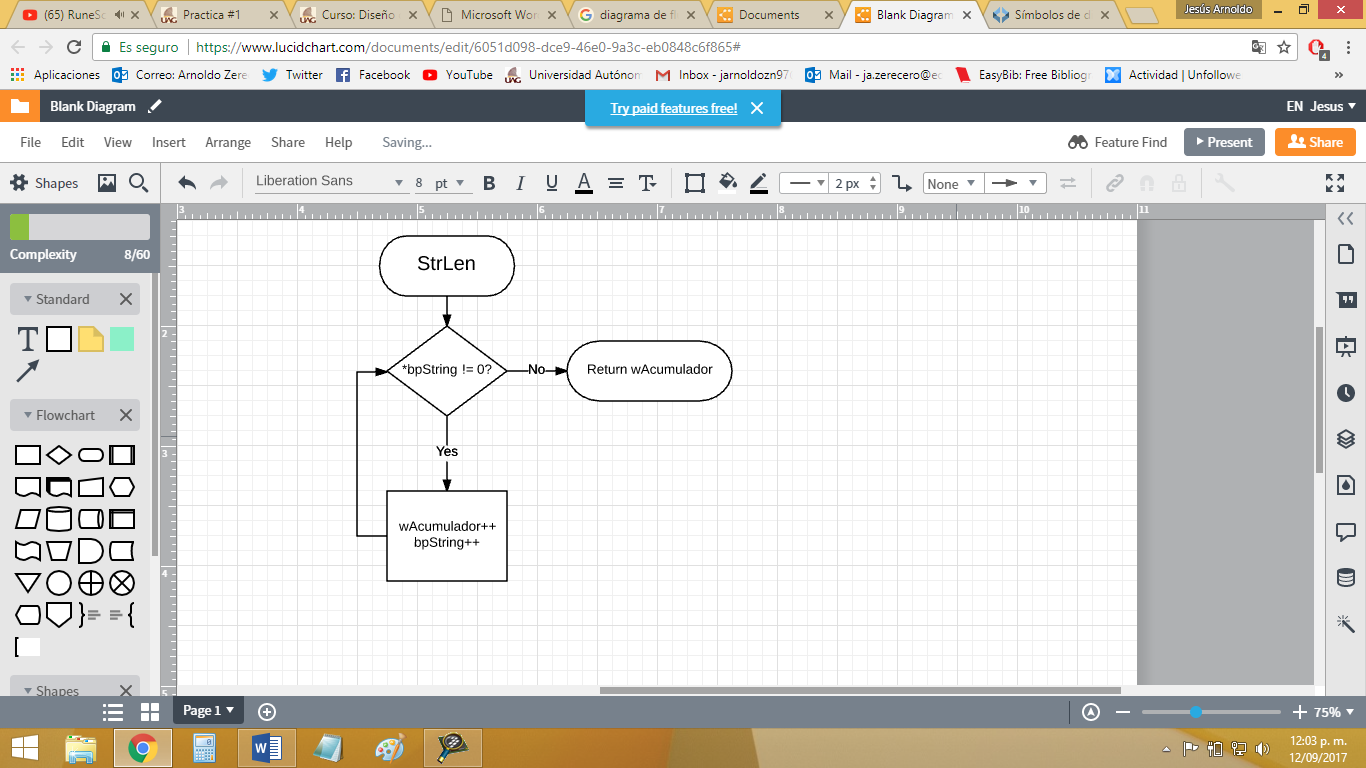
**ByteAddress:**



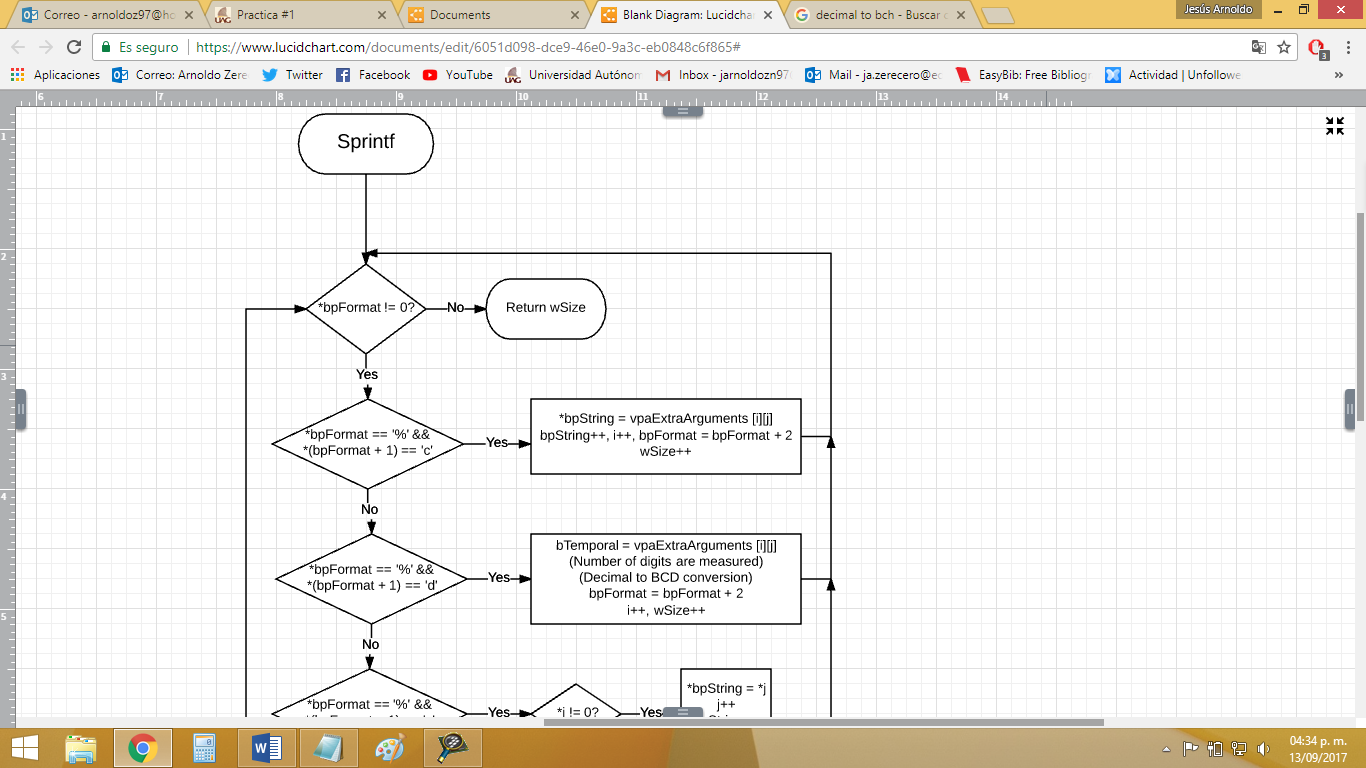
**Sort:**

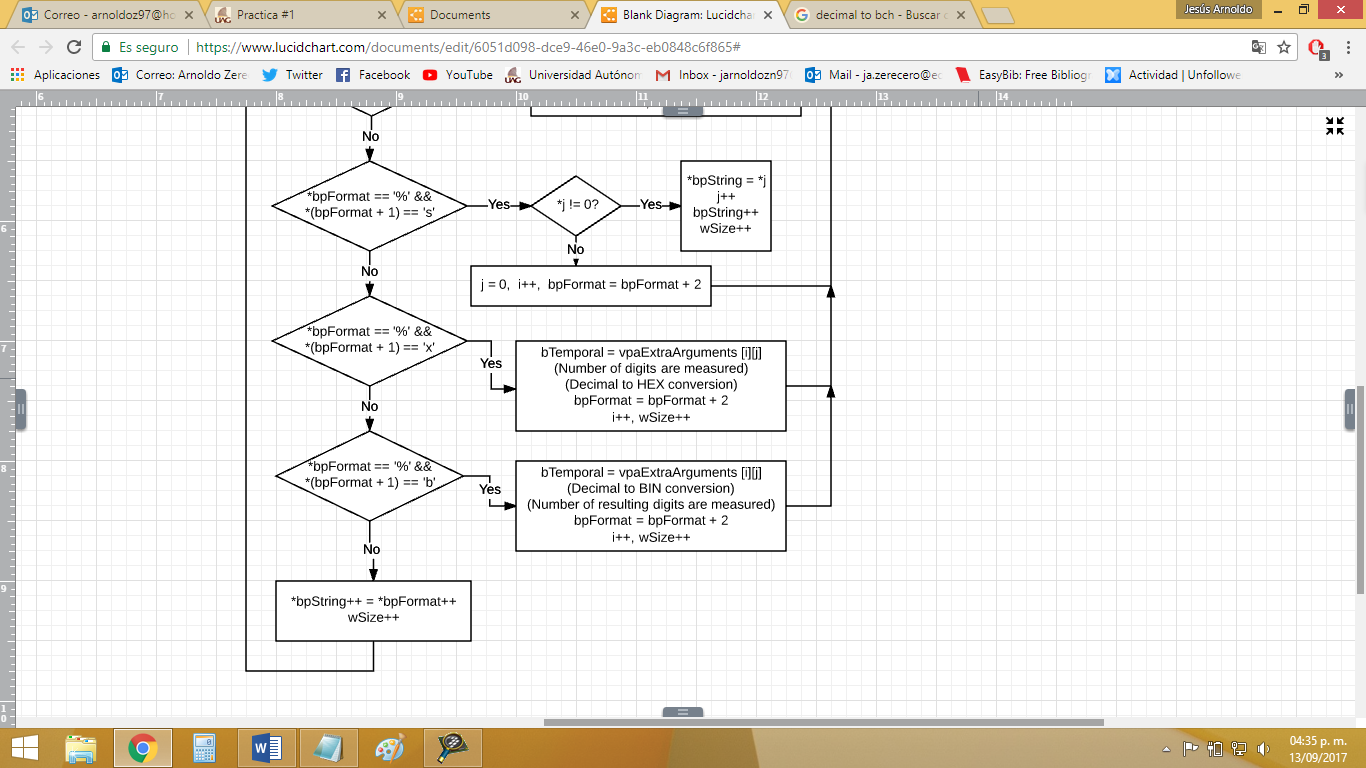


**StrLen:**



**Sprintf:**





**Conclusion:**

In conclusion, this practice helped me understand how these 9 functions work internally, so now I am able to understand more coding errors related to how I call or use these functions. Most of them were very simple to understand and recreate from scratch. The only problem I had was the sprintf function, which is, by far, the hardest. The problem with the sprintf function, apart from its length (as I had to code the conversion codes to %c, %d, %s, %x and %b) was retrieving and using the possible extra argument the user may send (…) without using the stdarg.h header or any of its functions. This was the hardest task, as it took days of thinking and researching to find a different way to approach this argument retrieving problem. The best approach was creating an array of pointers, in which the user may declare as many arguments as he pleases, while only needing to send the address of this array of pointers to the sprintf function.