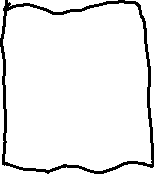
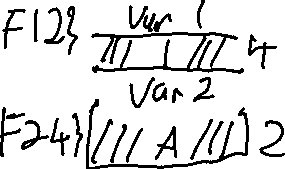
Programming – 12.11.19

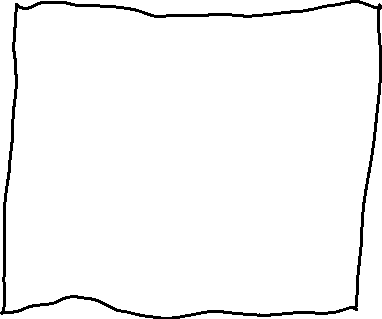
* Pointers: trickiest part of C
* Memory Management
* Only couple of languages support pointers, C, C++, etc
* Only allowed in lower level languages
* Allow the developer to manipulate/deal with a programs memory
* EX1 – Pointers example
* & < Address location
* %p shows the address of the variable; the datatype of the variable has no affect.
* %p of an array, will show the start of the array.
* There must be an & in front of the var, so &var1



* Memory map showing memory in relation to vars:



* The OS will NOT allow you to interact with memory it is using itself, this can be overwritten. But it is NOT recommended as this may lead to system instability or, in worst case, bricking.
* **A pointer variable is a variable that stores the memory address of another variable.**
* type \*pointer\_variable.name;
* int \*ptr.
* ^ A variable with a start infront of it means it is a pointer variable, meaning it will contain the memory address of some other integer variable.
* They MUST be the same datatype, you cannot store a char variable in a char pointer variable.
* \*ptr\_X = good practice for a pointer name, X being replaced with the desired name.
* EX2.c – pointers example
* Memory map describing the program:



* “ptrx = &varx” Example of “pointing” at a memory location.
* You can reassign a pointer later in a program, though you cannot change it’s type.
* Like any other assignment operation, the file location is assigned to the pointer variable as an address.
* You can only store a single address in a pointer, however you CAN have a string/array of pointers. Meaning they follow the exact same rules as any variable.
* The delimiter for a pointer is %p
* REMEMBER: The & is used to denote the memory location of the variable, meaning that when printing a pointer in a printf, you do not use the &. As this will show the memory location of the pointer itself, rather then what is contained within.
* %x will display the hex location, instead of the normal memory address.
* **AGAIN: Pointer variables follow the same rules as normal variables, this means you can equal them to each other, or do anything else with them you can do with normal variables, etc**