ICS 212 Homework #1

Part I

1. Use the divide-and-conquer method which we discussed in class to create a top-down-design drawing for a "throwing a party" simulator, using a structured approach. The approach should include three levels of decomposition, including the top level. There are many computer programs you may find helpful, for example Microsoft Word. Or, if you are using a PC, you can also download Microsoft Visio from MSDNAA free of charge. Whichever program you decide to use, please submit your drawing in either PNG, GIF, JPEG, PDF, or TXT format.

Part II

1. Create a new project called "GumBallMachine".

2. Within this project, create a header file called GumBallMachine.h. Within this header file, typedef a struct called GumBallMachineType. This type should have three fields, each of which will be ints: iNumGreenGumBalls, iNumRedGumBalls, iNumBlueGumBalls.

3. Create a corresponding source code module called GumBallMachine.c. Within this module, create a function called, "PrintGumBallMachineStatus()". This function should take one parameter, a structure of the above type. It should print a nicely formatted message describing the contents of the structure, e.g. what gumballs the gumball machine currently contains. The function should return an int, indicating success or failure (in this case, it is probably always going to return success). Remember to give the function a function comment block, using the style provided in class. Also remember to put the prototype for the function in GumBallMachine.h.

4. Create a module called main.c, and in this module put your function main(). In main(), define a single variable of type GumBallMachineType. After defining this variable, initialize each of the fields to reasonable values, and then call your function PrintGumBallMachineStatus() to print the status of your Gum Ball Machine. Remember that both in order to define variables of type GymBallMachineType, and in order to call function PrintGumBallMachineStatus(), module main() will need access to the contents of GumBallMachine.h, so be sure to include this header file.

5. Finally, for practice, try *dynamically allocating* a gumball machine. Do this within another function called, TestGumBallDynamicMemory() which you should put in GumBallMachine.c, and which you should call from main.c. This function should define a pointer to GumBallMachine, then call malloc() to request a block of memory sufficiently large to hold a gumball machine, then use the -> operator to record values within each of the fields, then use the -> operator again in conjunction with printf() to print the values of each of the fields, and then finally use free() to return the block of memory to the free memory pool.

Submitting

To submit your code, upload a ZIP file named *username*A01.zip to Tamarin, where *username* is your UH username. The filename format is case-sensitive. This ZIP file should contain your two .c files, your .h file, and your drawing file from Part I.