CS2261 Media Device Architecture - Recitation

* States:
  + All games have multiple states, like main menu, win, lose, etc
  + All states have three things in common
    - Set them up once
    - They do stuff (manage menus/ game mechanics, etc)
    - Transition to other states
  + State machine
* Enum {START, GAME, PAUSE, WIN, LOSE}
  + Just representing a number, 5 values that don’t currently mean a thing
* int state = START;
  + technically equals 0
  + represents the “Start” state right now
  + can check this variable to check what state we are in
* use switch statements instead of if-else chains because if-else is O(n)
  + switch is O(1), uses a jump table in memory, much faster
  + no need for a default state
* check the state each frame (every iteration of our main-while loop)
* State functions:
  + Put everything that is done each state in its own function
    - Start function – will repeatedly check every single frame if the start button Is pressed so it knows to switch to another state
  + There are something we don’t wanna do each state
    - Like filling the screen
      * If you call fill screen each frame, you have lots of flicker
    - Initialize
  + For each state, we will have a goTo function that you call at the start of transition to that state
    - goToGame() for example will only transition you to the game()
      * it changes the state to game so when the while loop runs again and the switch statement is hit again, the state is going to be game()
  + Have check-for-state transitions code at the END of the state’s function, put this last to avoid jumping to a new state before completing the state
* Have stuff like ifHealthIsZero = goToLose
* Randomness
  + Occasionally we want things to be different each time we play
    - In stdlib.h, we have a rand() function
    - Not actually random
  + Important equation for getting a random number in a range:
    - rand()%(max – min) + min
      * if you want to choose randomly between two numbers, like 4 and 10:
        + [4, 10)
        + Max – min = 6, add the min:
      * rand()%6 + 4
        + gets you [4, 9]
      * if your min is negative, do the same thing
    - say you want to choose a random element in a array
    - rand()%(LEN-min) + min
    - arr[rand()%ARRLEN]
  + problem is, the seed variable will always be the same, so the random numbers will always be the same and will appear in the same order every time you run your
  + to fix this, have the user control the seed
    - srand() – seed the random
* There is an initialize game function that initializes
  + if you call srand() before rand(), its use default seed (which is 1)
  + srand in stdlib.h
  + srand uses the seed for the random, and random uses that seed to make a “random” number
    - srand() generates a random seed but if its not used, the seed defaults to 1
    - rand() will then use the seed to generate a random file