Georgia Institute of Technology
School of Computer Science
CS 2110X: Spring 2014 (Conte)
Homework #5.
Making a Game on the Raspberry Pi

For this homework, you will be creating a game on your raspberry pi. You may pick one of the following options:

Snake	(http://en.wikipedia.org/wiki/Snake_(video_game))
Breakout	(http://en.wikipedia.org/wiki/Breakout_(video_game))
Missile Command	(http://en.wikipedia.org/wiki/Missile_command)
Tetris	(http://en.wikipedia.org/wiki/Tetris)
Asteroids	(http://en.wikipedia.org/wiki/Asteroids)

You do not have to follow the look and feel of these games exactly - feel free to add your own flair - but they should be recognizable and have the same goal. **We will award extraordinary games with extra credit of up to +10%.** You are free to implement your own creative game idea instead, as long as you email Scott with the details of the game for approval first.

The first step of making your game is implementing all empty functions in main.c. You may (and should) add any other additional helper functions you feel are necessary.

Once the functions are implemented and working correctly, you may start working on your game. We have provided template code for you to use in main() - you probably do not want to remove what's there! **All games must include the following:**

- A start screen / title screen. It would be helpful if you put instructions for your game here.
- A visible score counter or some other text
- Keyboard input
- Winning / losing condition
- Images included in gameplay
- At least 5 different colors
- Collision detection
- At least one struct to hold game object information
- Ability to be paused
- Game over screen
- A README.txt file describing your game, how to play, any features you're particularly proud of, etc.

To submit, 'make clean' then make a tarball of the entire folder containing your game and README.txt (tar -zcvf hw5-lastname.tar.gz ./hw5/). **You must submit everything needed to successfully compile your game!**

In order to help with completing this assignment, we have provided a simple library of functions for common tasks in gamelib.h and text.h:

Keyboard Input:
You must install ncurses on your raspberry pi (sudo apt-get install libncurses5-dev)
getKeyPress() [gamelib.h / gamelib.c]
Returns an int corresponding to the last character the user entered, or -1 if nothing was entered. Calling getKeyPress() consumes the character from input, so you probably do not want to call this function multiple times in your game logic loop. The int returned can be any ascii character ('a', 'D', '8', '!', '\n', although for a game you probably only want lowercase letters), as well as some special inputs (KEY_LEFT, KEY_RIGHT, KEY_UP, KEY_DOWN, KEY_BACKSPACE, see http://www.gnu.org/software/guile-ncurses/manual/html_node/Getting-characters-from-the-keyboard.html for the full list).

Drawing Text:
charHasPixelSet(c, i, j) [text.h]
A macro used to determine whether to turn on pixels or not when printing a character. This macro looks up 13x7 monochrome character bitmap information in the text_bits[] array defined in text.c. For example, if you wanted to know whether the bottom left pixel of the character 'a' was set, you would call 'if (charHasPixelSet('a', 12, 0)) { … }'.

fontsize_t [gamelib.h]
An enum that lists out the font sizes you should implement and use when drawing strings and characters. You can use these just like ints:

```
fontsize_t fontsize = LARGE;
for (int i = 0; i < fontsize; i++) { . . . }
```

The number next to the name in the enum specifies its scaling factor - TINY text is 1:1 bitmap text (each bit in the bitmap corresponds to one pixel on the screen), MEDIUM text is 3:1 scaling (each bit in the bitmap should map to 3x3 pixels on the screen), etc.
Hint: you do not need separate code for each font size!

Drawing Images:
Install GIMP (http://www.gimp.org/)
(Feel free to do this in Windows, or sudo apt-get install gimp on your raspberry pi).
To use an image in your game:
1. Scale it to the appropriate size (Image -> Scale Image)
2. File -> Export. Save as [imagename].c
4. In the resulting window, check "Save as RGB565 (16-bit)" and nothing else. Change prefixed name to [imagename]. Click Export.
5. Move the file into your game directory, and add #include "[imagename].c" to main.c
(note: including c files is bad practice, but due to how GIMP exports the image, this is the easiest way for now).
6. Access image width, height, and pixel_data in main.c as you would with any other struct.