LAB 11:

Linked List

Provided Files

- main.c
- mode0.c
- mode0.h
- spritesheet.c
- spritesheet.h
- list.c
- list.h
- tiles.c
- tiles.h

Files to Edit/Add

- main.c
- list.c
- Makefile
- .vscode
    - tasks.json

Instructions

In this lab, you will be implementing a linked list in order to complete a classic game of snake! Each linked list node represents a segment of the snake, therefore we must be able to create a linked list, create nodes, push them to the back, and update node positions.

Note: Make sure to copy over your Makefile and .vscode/tasks.json from one of your previous assignments.

TODO 1 – Setting Up the Linked List

First, let's set up our linked list functions! Since our functions to create a linked list are not complete, if you build and run at this time you will only see a food object. Navigate to list.c.

TODO 1.0

- Complete the create_list function. This function should create a doubly-linked list, using the LIST struct. This new list should have both head and tail initialized to NULL.

- **NOTE**: Make sure to handle the case where malloc cannot allocate enough memory for the list.

TODO 1.1

- Complete the create_node helper function. This function should create a node of type NODE by allocating memory for it on the heap. The parameter pos, of type POSITION, should be stored in the node. Be sure to set its pointers to NULL.

TODO 1.2

- Complete the push_back function. This function should add data to the end of the linked list. To do this, it should create a node storing the pos parameter, and then add that to the front of the dllist parameter.
    - **NOTE**: Make sure to handle the case where the dllist parameter is an empty list.
    - **HINT**: Feel free to reference the push_front function, which adds the data to the front of the linked list.

Build and run. At this point, you should be able to see the moving snake head and four snake segments that are not moving, as well as the food object. The moving snake part should be controlled by using the UP, DOWN, LEFT, and RIGHT buttons. If the moving part collides with any of the body segments or with the edge of the screen, the screen should be filled with tiles; at this point, you can click the START button to restart the game.

TODO 2 – Updating Node Positions

Now you'll focus on an essential function for our snake game: updateNodePositions. This is what is going to make the snake's body segments move with the snake's head.

TODO 2.0

- Complete the updateNodePositions function. This function takes in a pointer to the list which contains the nodes we want to update. The purpose of this function is to start at the end of the list and set each node's position to the previous node's position.

Build and run. You should be able to play the game in its entirety at this point. Using the same controls listed above, the snake's body should be fully connected, and eating a food object should add another node to the linked list, expanding the snake by one tile.

You will know if it runs correctly if:

- You can play a game of snake without any weird issues!

Tips:

- Follow each TODO in order, and only move forward if everything is correct.

- Reference list.h to see the structs you should be using for this!

Submission Instructions:

Ensure that **cleaning** and building/running your project still gives the expected results. **Please reference the last page of Lab02.pdf for instructions on how to perform a "clean" command.**

Zip up your entire project folder, including all source files, the Makefile, and everything produced during compilation **(including the .gba file)**. Submit this zip on Canvas. Name your submission Lab11_LastnameFirstname, for example:

“Lab11_CobraKing.zip”

It is your responsibility to ensure that all the appropriate files have been submitted, and that your submitted zip can be opened and everything cleans, builds, and runs as expected.