UCP Assignment – Turtle Graphics Program
Jhi Morris (19173632) – 19/10/18

Part 1. Function Descriptions

main.c:

main(int, char*): Calls multiple other functions, checks that they completed without error, and prints appropriate error messages in the case of a fault.

freeList(LinkedList): Frees the linked list and contained linked nodes & values.

turtle.c:

parseCommands(LinkedList, int): Tracks state of the turtle, reads strings in the linked list and passes them to checkCommand() for parsing, then acts on that information to parse the following arguments and execute the commands to update the state of the turtle and plot on the screen using line(). Invalid parameters are ignored if strict mode is disabled, otherwise the function is quickly exited and an error code is returned.

checkCommand(char*): Creates a copy of the string, then compares it against an array of constants relating to known commands.

rotate(TurtleState*, char*): Gets double from command argument, executes rotate command updating turtle state.

move(TurtleState*, char*): Gets double from command argument, executes move command updating turtle state.

draw(TurtleState*, char*): Gets double from command argument, executes draw command updating turtle state and drawing to the screen.

fg(TurtleState*, char*): Gets int from command argument, executes FG command if not simple.

bg(TurtleState*, char*): Gets int from command argument, executes BG command if not simple.

pattern(TurtleState*, char*): Gets char from command argument, updates turtle state.

toUpper(char*): Converts all lower-case letters in the string to upper-case.

roundDouble(double): Rounds a double to the nearest integer.

fixAngle(double): Wraps an angle to be between 0 and 360 degrees.

move(double*, double*, double, double); Updates the x/y coordinates according to the angle of direction and magnitude of movement.

plotting(void*): PlotFunc; prints the handed character to the screen at the cursor.

fileio.c:

loadLines(char*, LinkedList): Loads the file at the filename, exports a linked list of nodes containing values of each line of the file as strings.

allocateLineNode(LinkedListNode**): Allocates memory on the heap for the node to store line strings and initializes ‘next’ for comparison.

makeLog(char*): Opens an appending filestream the first time run. Writes the string to the file. Closes the file when run with an empty string.

Part 2. On converting input files to coordinate systems

After the input files are read line-by-line into the back of a linked list the first word of them is compared against an array of known strings relating to the supported commands. Once the command has been identified the arguments for it can be read in.

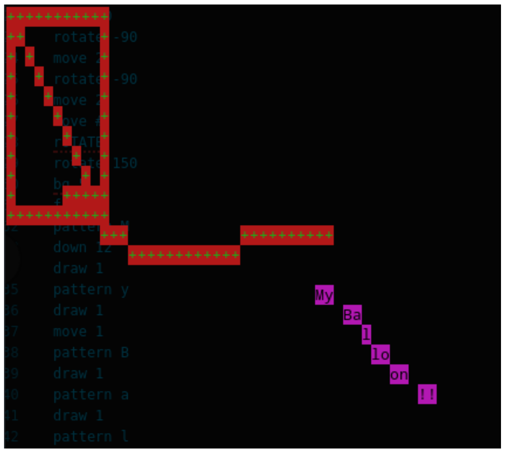
Some commands require characters as arguments, other floats or integers. This is why the command word is parsed in a separate step prior to the arguments.

Once the command and arguments are parsed, the relevant operations can be executed. This is done by updating variables contained in the TurtleState struct which is then passed to the relevant function along with the string from the linked list node. In the case of ROTATE this value may need to be adjusted (to stay within 0-360 degrees). In the case of DRAW this includes keeping track of the previous unadjusted coordinate and passing both the new and old coordinate sets to the line() function.

An alternative to this approach would be to do parsing at the file reading step, instead creating a linked list of structs containing turtle command data. This moves the responsibility of parsing the commands away from the procedure of executing the commands, which may improve portability. This approach would likely require an additional step during processing of commands where the structs are differentiated from each other, practically requiring an additional pass of parsing. If the process of file-reading and command-executing was combined then this additional step would be made redundant, but this would reduce portability and cohesion.

Part 3. Example input & output

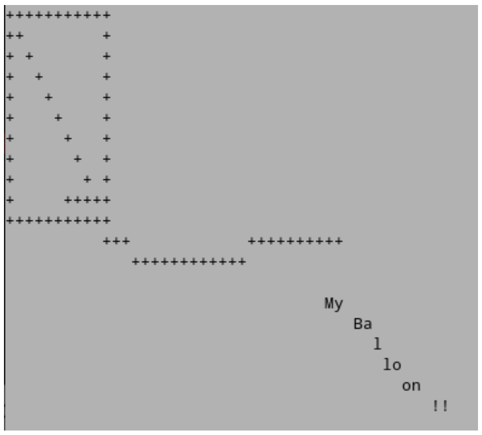
```
~$make all
~$./TurtleGraphics test.txt -s
>Loaded 52 lines from file “test.txt”.
>Unexpected line in input file. Ending because of -s flag.
~$./TurtleGraphics test.txt
>Loaded 52 lines from file “test.txt”.
```



```
+++++++++++
++        +
+ +       +
+  +      +
+   +     +
+    +    +
+     +   +
+      +  +
+       + +
+     +++++
+++++++++++
          +++          ++++++++++
             ++++++++++++

                                 My
                                    Ba
                                      l
                                       lo
                                         on
                                            !!
```

```
>Drawing complete with 6 errors.
~$./TurtleGraphicsSimple test.txt
>Loaded 52 lines from file “test.txt”.
```

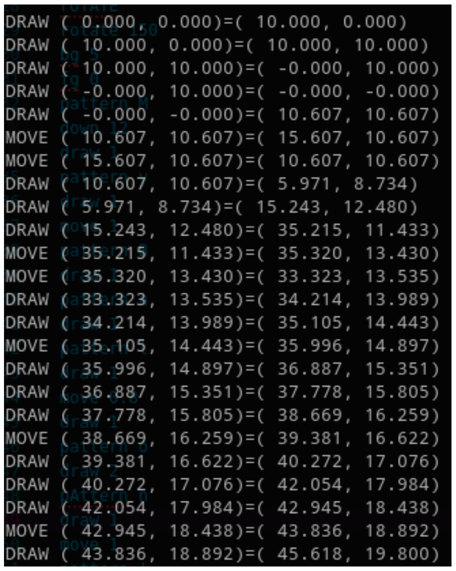


```
+++++++++++
++        +
+ +       +
+  +      +
+   +     +
+    +    +
+     +   +
+      +  +
+       + +
+     +++++
+++++++++++
          +++          ++++++++++
             ++++++++++++

                                 My
                                    Ba
                                      l
                                       lo
                                         on
                                            !!
```

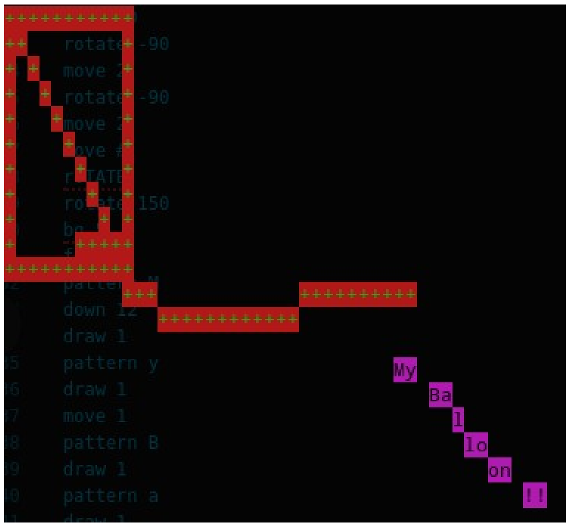
```
>Drawing complete with 6 errors.
```

```
~$./TurtleGraphicsDebug test.txt
>Loaded 52 lines from file “test.txt”.
```



(Output via stderr)

```
DRAW ( 0.000, 0.000)=( 10.000, 0.000)
DRAW ( 10.000, 0.000)=( 10.000, 10.000)
DRAW ( 10.000, 10.000)=( -0.000, 10.000)
DRAW ( -0.000, 10.000)=( -0.000, -0.000)
DRAW ( -0.000, -0.000)=( 10.607, 10.607)
MOVE ( 10.607, 10.607)=( 15.607, 10.607)
MOVE ( 15.607, 10.607)=( 10.607, 10.607)
DRAW ( 10.607, 10.607)=( 5.971, 8.734)
DRAW ( 5.971, 8.734)=( 15.243, 12.480)
DRAW ( 15.243, 12.480)=( 35.215, 11.433)
MOVE ( 35.215, 11.433)=( 35.320, 13.430)
MOVE ( 35.320, 13.430)=( 33.323, 13.535)
DRAW ( 33.323, 13.535)=( 34.214, 13.989)
DRAW ( 34.214, 13.989)=( 35.105, 14.443)
MOVE ( 35.105, 14.443)=( 35.996, 14.897)
DRAW ( 35.996, 14.897)=( 36.887, 15.351)
DRAW ( 36.887, 15.351)=( 37.778, 15.805)
DRAW ( 37.778, 15.805)=( 38.669, 16.259)
MOVE ( 38.669, 16.259)=( 39.381, 16.622)
DRAW ( 39.381, 16.622)=( 40.272, 17.076)
DRAW ( 40.272, 17.076)=( 42.054, 17.984)
DRAW ( 42.054, 17.984)=( 42.945, 18.438)
MOVE ( 42.945, 18.438)=( 43.836, 18.892)
DRAW ( 43.836, 18.892)=( 45.618, 19.800)
```



(Output via stdout)

```
>Drawing complete with 6 errors.
```

“test.txt” can be found attached along with this report. The important faults that it attempts to check for include invalid command words, negative angles for ROTATE, negative magnitudes for MOVE/DRAW, invalid values for FG/BG, missing command arguments and empty lines.