asgn4: The Perambulations of Denver Long

Program is divided up into 4 .c files and 4 .h files. vertices.h, graph.c/.h, stack.c/.h, path.c/.h, tsp.c.

vertices.h:
Extrememly simple file provided by Long that only holds a defined value for START_VETEX and VERTICES.
These two values are used in later .c files as psuedo-global variables so that they can easily be changed at will.

tsp.c:
Acts as the main file that allows the user to interact with the implemented graph API and starts the DFS algorithm.
Allows the user to choose a specified input file to create a custom graph and choose the output file to print the output
.
Start:
Use getopt to check which characters the user inputted when running the program
-h : Help message. Prints the usage of the program
-v : Verbose Printing. Prints each new optimal path as it is found
-u : Uses an undirected graph
-i infile : specifies the input file
-o outfile : specifies the output file
Run DFS on the given graph to try and find the shortest possible path that connects every single vertex.
DFS runs by searching vertices on a sort of first come first serve basis, where the next vertex that DFS will search is

determined by the lowest value of each of the existing vertices that it has not yet searched. Only once it has searche
d all available
vertices from a given vertex does it retreat to the previous vertex and restart the cycle of exploring vertexes.
Print the shortest Hamiltonian path found and the path itself to the given output file.

graph.c:
Acts as the actual meat and potatoes for the graph API.
Has the ability to
-create a new graph with the requested number of vertices and requested directionality
-sets all graph variables to default values
-delete given graph
-add an edge to existing graph
-alters the matrix array in the graph object to include the given edge weight at specified position
-return certain pre-existing variable statuses including: bool has-edge, edge-weight, bool visited
-mark a vertex as visited/not visited
-alter the visited array at specified vertex to mark as visited/not visited

stack.c:
Houses all of the code for the stack API.
Acts as a normal stack with the exception that the values stored in the stack are in fact stored in an array
with all of the stack code simply acting as the front for interacting with the array as if it were a stack.
Has the ability to
-create a new stack with the specified capacity
-delete a given stack
-return several variable values including: bool stack-is-empty/full and stack-size
-push and pop off of the stack
-makes use of the top variable to keep track of the stacks position in the array then simply
adjusts the arrays value accordingly
-peek at the stack
-simply returns the value on top of the stack without modifying said stack

path.c:
Stores the path API needed to actually search through a graph with Depth-first search.

The path API in essence is just comprised of a stack and an int to store the length of the path, so not too much is actually done in terms of computing and processing, simply storing neatly in a stack.
Has the ability to
-create a path
  -set size to default value of zero
-delete a given path
-push and pop a vertex off of a path
  -simply adds/removes the vertex onto the stack and properly adjusts the size variable
-return several existing variable values including: number of vertices, and path length