**Assignment 2 – Pathfinding and Alpha-Beta Pruning**

**CISC 352**

**Due March 16, 2018**

**Spencer Edwards - 10129952**

**Hannah LeBlanc - 10091837**

**Pathfinding**

**Alpha-Beta Pruning**

The program as a whole is controlled by the solvePrunes function. This function reads the input file “alphabeta.txt” containing the specifically formatted sets of nodes and edges, creates a Graph class for each graph it reads in (a graph consisting of one set of nodes and one set of edges), and then write the output requested by Professor Kaos to the file “alphabeta\_out.txt”.

The Graph class is designed to setup a single graph/tree that it then traverses using the MiniMax algorithm in combination with alpha-beta pruning. This class really consists of two main parts; the parsing of the nodes and edges into their respective data structures, and the traversal of the graph.

The parseNodes function takes the string representing the set of nodes as read from the input file, and parses the string into pieces to ultimately form a Python dictionary (which are implemented more or less as hash tables). In this case, the key for a dictionary entry is the letter of the node (ie. “A” or “Abc” depending on the node), and the value for the entry is either “MIN” or “MAX”, depending on if that particular node is at a layer of the tree that is a max layer or a min layer.

The parseEdges function works very similarly to the parseNodes function. The parsing works in a very similar manner, but the dictionary is set up differently. The key for a dictionary entry is once again the letter of the node, but this time the value is actually a list of other nodes to which the key node is connected to via edges. The list of edges of a key node contains just the children of that node (in other words, the parent of the key node is not in the list).

The alphaBeta function pretty much follows the algorithm given by Professor Kaos in both the course notes and the assignment description. It is obviously tailored to the way it was implemented in our code (ie. our choice of data structures, etc), and really just differs in manner by which we must examine leaf nodes. We do not actually store the leaves as nodes, as they are instead just stored in the dictionary of edges for a “true” or “letter” node. If a node has leaves, the leaves are iterated through via a loop instead of recursively calling the alphaBeta function on them as is done for true nodes. While efficiency was not a primary concern for this assignment given that it is for the most part implemented recursively, eliminating the need to call the alphaBeta function on the leaf nodes as well as the true nodes should speed things up a bit. Other than this, the alphaBeta function follows the algorithm described in the course notes and assignment description.