1 Introduction

In this lab you will be extending the 2D Linked Lists data structure started in class.

1.1 The Unseen Start Pointer

As mentioned in class, there is an unseen pointer that points to the top-most `node2D`, the node with data `joe` in the example I did. If you think of the `node2D`'s in the left column as a linked list, this unseen pointer would be the head pointer of that linked list.

You should declare an external (i.e., outside all functions) variable used for this purpose. I will call it "the unseen start pointer" in the assignment, you may call it whatever you wish. Initially this pointer is `NULL`.

1.2 Overview of Commands

Your program reads commands from the terminal. A few commands cause something to be printed; others only affect the internal state.

All command names are designed to be read with `getword()` from section 6.3 of the text and class notes. Most commands take one or two arguments, which are strings (they are the names of various nodes). These names can all be read with `getword()`. Recall that our version of `getword()` (found in the text and notes) requires `getch()` and `ungetch()` from the text/notes.

The commands are not command-line arguments. All command names as well as all node names are guaranteed to be no longer than 20 characters. You can depend on this assumption and do not have to check if it is violated.

Details about individual commands are in the next section In this subsection we just give rough descriptions and general comments.

Many commands create new nodes (1D or 2D). All 2D nodes are the same size (namely `sizeof(struct node2D)`) and all 1D nodes are the same size (`sizeof(struct node1D)`). You should call `malloc()` whenever you create a new node. Indeed, as discussed in class, `malloc()` will often be called several times: once for the node and once for the each name in the node.

As mentioned above, you may assume that the name of any node is no longer than 20 characters.
Specifically, `strlen(name)<=20` for any 1D or 2D node.

1.2.1 Append 2D Node

Create a new `node2D` and append it at the end of the (vertical) list of 2D nodes.

1.2.2 Append After 2D Node

Create a new `node2D` and append it after an already present 2D node.

1.2.3 Append 1D Node

Create a new `node1D` and append it at the end of the specified 1D list.

1.2.4 Append After 1D Node

Create a new `node1D` and append it after an already present 1D node.

1.2.5 Print All

Print the entire diagram in the style shown in class.

1.2.6 Print 1D List

Print the 1D (horizontal) list headed by the `node2D` given in the command.

1.2.7 Find 1D Node

Search for a 1D node with the `name` given in the command. Print "found" or "not found" accordingly.

2 The Commands in Detail

2.1 Commands That Add Nodes

2.1.1 Append 2D Node

Create a new `node2D` and append it at the end of the (vertical) list of 2D nodes.

The format of the command is
`appendRear2D name`

The 2D node you create and append will have this `name`. It's `first` and `down` pointers will be `NULL`.

You may keep a pointer to the end of the vertical list so that you can append in constant time, which is asymptotically optimal. You may also always begin at the unseen start pointer and keep following the `down` pointers until you reach the end of the list. Although this second method is asymptotically worse, it might be easier to implement and you will not be penalized for using it.

2.1.2 Append After 2D Node

Create a new `node2D` and appends it after (i.e., below) an already present 2D node.

The format of the command is
`appendAfter2D name afterName`

The 2D node you create will have this `name` and it will be placed just after (i.e., below) the existing 2D node having name `afterName`.

If there is no existing 2D node having name `afterName`, print an error message and abort the run.

If there are several existing 2D nodes having name `AfterName`, you may use any one of them.

2.1.3 Append 1D Node

Create a new 1D node and append it as the last node of the specified 1D list

The format of the command is
`appendRear1D name name2D`.

The 1D node you create will have this `name` and will becomes the last (i.e., rightmost) 1D node on the (horizontal) list of 1D nodes headed by the 2D node having name `name2D`.

If there is no existing 2D node having name `name2D`, print an error message and abort the run.

If there are several existing 2D nodes having name `name2D`, you may use any one of them.

2.1.4 Append After 1D Node

Create a new 1D node and append it after an already present 1D node.

The format of the command is
`appendAfter1D name afterName1D`.

The 1D node you create will have this `name` and will immediately follow (i.e., will be immediately to the right of) the existing 1D node having name `afterName1D`.

If there is no existing 1D node having name `afterName1D`, print an error and terminate the run.

If there are several existing 1D nodes having name `afterName1D`, you may use any one of them.

2.2 Commands That Print Diagrams

2.2.1 Print All

Print the entire diagram in the style shown in class and illustrated in the class notes.

The format of the command is
`printAll`

2.2.2 Print 1D List

Print the 1D list headed by the node2D given in the command.

The format of the command is
`print1DList name2D`

Find an existing 2D node having name `name2D` and print its associated 1D list. That is, the name of the 2D node is printed as are the names the 1D nodes on the (horizontal) list headed by this 2D node.

If there is no existing 2D node having name `name2D`, print an error message and abort the run.

If there are several existing 2D nodes having name `name2D`, you may use any one of them.

2.3 Other Commands

2.3.1 Find 1D Node

Search for a 1D node with the given name and print "found" or "not found" accordingly.

The format of the command is
`find1D name1D`

2.3.2 Find 2D Node

Search for a 2D node with the given name and print "found" or "not found" accordingly.

The format of the command is
`find2D name2D`