Global variables:

- Declare a stack variable
- Create a string variable for the infix input
- Create a string variable called postfix

Main.cpp

- Include the header files and cpp files for all the classes that are created
- Call the input function
- Create an instance of the Node class
- Call the concert function on the Node object
- Output the postfix string

Class header and .cpp files

- Node.h
    - Node Class defined here
        - Private:
            - Initialize a string variable for operator
            - Initialize a string variable for operand
            - Initialize a next pointer as an instance of the class.
        - Public:
            - Accessors:
                - Declare a precedence function
                    - Argument is a string
                    - Return an int
                - Declare a bool operator check function
                    - Argument is a character
                    - Return true or false
            - Mutators:
                - Declare a convert function
                    - Return a string
                    - No function argument

- Constructors:
    - Create a default empty constructor

- Node.cpp
    - Node class accessors,mutators, and constructors defined here
        - Precedence function with (::)
            - Argument is a string
            - Declare an integer variable
            - If string is equal to other an Asterisk (multiplication), or Division
                - Integer variable is equal to two
            - Else
                - If string is equal to either a plus or a minus
                    - Integer variable is equal to one
            - If string is equal to a period
                - Set integer variable to zero
            - Return integer variable
        - Operator check function with (::)
            - Argument is a character
                - If the character is equal to any of the operators listed in the convert function cases below
                    - Then return true;
                - Else
                    - Return false
        - Convert function with (::)
            - No argument
            - For (counter is equal to 0, counter is less than infix string .length, counter increment by 1)
                - Switch infix at index i
                - Case +
                - Case -

- Case *
- Case /
- Case ^
- Case (
- Case)
- Case .

- Set operate equal to infix at index i
  - If the global stack variable is empty or precedence (operator) is greater than or equal to the precedence at the top of the stack
    - Stack variable.push function is called
      - Argument is the string operator
      - Break;
  - Else
    - While stack is empty and precedence of the operator is less than the precedence of the stack.top_stack function,
      - Append the positfix string
        - Call stack.top_stack function as the argument
      - Pop the stack
  - Push the stack with the argument of the string operator
- Default: (when operand string char is parsed)
  - Operand is set equal to infix at index i
  - Append the postfix string (argument is the operand)
  - break;
- While stack is not empty
  - Append the postfix string

                - Call the top_stack function as the argument
            - Pop the stack
        - Return postfix string
        - Output the postfix string
- Stack.h
    - Stack class defined here:
        - Private:
            - Initialize an int for size
            - an instance of the Node class to a pointer: top;
            - Inline function called stack with size set to 0 and top set to NULL
        - Public:
            - Declare function additem
                - Return type void
                - Attribute char or int
            - Declare pop function
                - No argument
                - Return type string
            - Declare get_size function (In line function)
                - Return the integer size
            - Declare push function
                - Argument is a string
                - Return string
            - Declare stack_upper() function
                - Return type void
            - Declare stack_empty function
                - No argument
                - Return a bool
            - Declare a top_stack function
                - No argument

        - Return a string
    - Declare a recursive print function
        - Return type recursive
- stack.cpp
    - Stack accessors,mutators, constructors, and destructors defined here
        - Create a Constructor for the class
            - Initialize all private variables of the stack class to 0 or NULL depending on what the variable is
        - Additem function with (::)
            - Dynamically create a pointer object instance of the node class
            - Pointer->data = char or int
            - Pointer next equals first
            - First equals Pointer
        - Recursive print function (::)
            - No argument
            - create a pointer object instance of the node class
                - Pointer equals first
            - If pointer is not equal to null
                - Output pinter->int or char
                - pointer = pointer->next
                - Call the function again with pointer
        - Top_stack function (::)
            - Return top - this could be an inline function
        - Push function (::)
            - Node class object pointer called temp
            - Temp points to a dynamically created instance of node class
            - If top is not equal to NULL
                - Temp next is equal to NULL
            - Else

        - Temp next is equal to top
      - Temp data (int or char) is equal to the string argument of the function
      - Temp equals top
      - Increment size by one
  - Stack_empty function
      - No argument
      - If the return of get_size function equals zero
        - Return true
      - Else
        - Return false
  - Pop function
      - Declare a string variable
      - If stack_empty function is true or exists
        - Output something that says that the stack is empty
        - Return
      - Else
        - Initialize a Node class object pointer called temp
        - Set temp equal to top
        - Top moves to the next item in linked list
        - String variable equals temp->data
        - Delete temp
        - Decrement size by one
      - Return the string variable
      - This is a destructor that recursively deletes the linked list

Other functions that are going to be used:

- Input function:
  - Ask for the infix string
  - Return infix string