Book

Book(); These will create the book object assigning basic values

~Book(); Deletes all extraneous pointers and arrays

String reduceWords(string); this will make sure that we do not get irregular words in our index

Take word

- Remove special characters
- Keep lower case characters
- Convert upper to lower

String toLowerCase(string); This will convert words to lower case for various reason

Take word

- Remove special characters
- Convert upper to lower

Void createIndex() this will create the index

Call create html func

Void createBook(instream); this will create the book based on the ASCII text file

While Reading file

- Split line into words
- Reduce words
- Add words/location to index
- Determine type of page
- Generate html

Void generateStopList(); this will generate the stop list so that we get the right index words

While reading file

- Add word to stop list linked list

Void setTitle(istream); this will set the title so all the pages have it

While reading file

- Split line into words

To lower case

Check if line is title:

Add everything preceding till newline as title

Index

Void createHtmlFile(string, int) This will create the index page based on all of the inputted words

Generate alpha index based on first characters in word

While head != null

    Check if occurs to many times or in stop list

    Add word and link to all locations of said word

Page

creatHtmlPage(); This will create said html page for the book

determine format of page name

write title

figure out format for next, prev and whether or not to show next or previous

write link to index page

write page content as added

cleanup

Words

Words(); Constructs linked list object

~Words(); removes all references to pointers

Void addWordOrLocation(string, int); figures out if a word needs to be added or its location

Call find word

    Add location if word is found

    Else add word in sorted order

Bool inStopList(string, Words); checks if a word is in the stop list passed in

While head != null

Check if word is in stopList

Return value

ListNode* findWord(string); finds the word in the current linked list or returns null

While head != null

Check if the word is contained in linked list

Return value or NULL

Word(string int); declares a new word variable

Bool locationAdded(int); checks if the location is already added to the word object

Look through vector for location

Return value;