CS 121 Project 2: Sets - Design

Jackson Staples

February 12, 2024

1 Data Structures

- Struct
  - Includes a string as data
  - Pointer to next node

2 General Function

Initialize two Node pointers as NULL for each of the selected text files. To read in the selected text files and create linked lists with each word as a string for the data in a node. Then the program takes the two linked lists, and find the Intersect and the Union of the two linked lists, creating a list for Intersect and Union respectively in the process. Then print the Intersect and Union.

3 Function Prototypes

- void readfile(Node*&, string)
- void printlist(Node*)
- void appendtolist(Node*&, string)
- bool checklist(Node*&, string)
- bool isAlpha(char)
- Node* addlists(Node*&, Node*&)
- Node* findintersection(Node*&, Node*&)
- Node* findUnion(Node*&, Node*&)

4 Functions

- readfile(Node* &head, string z)
  - Arguments:
    - Pass by reference Node head
    - Filename as a string
  - Iterates through the text file using Ifstream
  - Passes text to isAlpha() and checklist() to remove alphanumerics
  - If string passes checklist(), it gets sent to appendtolist()
- alphanum(string stringMem)
  - Arguments:
    - Create string buffer
  - Creates loop for each char in stringMem
  - If char passes through isAlpha(), add it to buffer string
- isAlpha(char temp)
  - Arguments:
    - Pass in a char
  - Checks char with ascii values, returning true if char is an alpha, hyphen, or single apostrophe
- printlist(Node* head)
  - Arguments:
    - Node pointer
  - Return error message if list is empty
  - Else, iterate through linked list, and print each node along the way
- appendtolist(Node*& head, string newData)
  - Arguments:
    - Node pointer
    - string
  - Sets data for node = newData
  - Creates new node member of the pointer that was passed to the function, with the string as its data member.

- checklist(Node*&, string search)
  - Arguments:
    - Node pointer
    - string
  - Create node pointer
  - While node != NULL, check if its data matches string search
  - If search matches data, return true
  - Set p equal to next
- addlists(Node*& S1Head, Node*& S2Head)
  - Arguments:
    - Node pointer
  - Pass in pointers for linked lists from both selected text files
  - Initialize node pointer Union to NULL
  - Set pointer p = S1Head
  - While p != NULL, add data from p to new linked list "Union"
  - Repeat same process for S2Head
- removeNode(Node*& head, string search)
  - Arguments:
    - Node pointer
    - string
  - Set new pointer P to head to check first item in linked list
  - If data from head == search, skip next, delete node, return to exit loop
  - Else, iterate through list with pointer mNode = p -⟩next
  - Skip next, set previous node to NULL, delete node, return to exit loop
- findintersection(Node*&, S1Head, Node*&, S2Head)
  - Arguments:
    - Node pointer
  - Initialize new node pointer, intersect as NULL

  - Set Node pointer p = S1Head
  - Iterate through list, set string search = data from S1Head, use checklist() to see if search == data from S2Head. if yes, then use appendtolist() and add search to linked list intersect.
  - return intersect

- findUninon(Node*&, intersect, Node*& Union)
  - Set new pointer p to intersect
  - Iterate through list, setting string search = data from intersect.
  - Pass search and Union into removeNode()to see if search is a member of Union; if yes, remove that node.
  - return Union