ADVANCED DATA STRUCTURES

PROJECT REPORT

NAME:	Advait Ambeskar
UFID:	9615-9178
EMAIL:	ambeskaradvait@gmail.com

Input

Text File with patterned input divided mainly into 3 patterned sequences.

1. Tree input- Begins with “$” followed by the keyword followed by the value
2. Tree output- Begins with the value (number of keywords to be returned)
3. Stop Operation- Begins with stop.

The text file is accessed through its location provided at the beginning of the execution.

Output

Text File with list of keywords, each new list on a separate line.

Class Declaration

We use three classes for the scope of this project.

The three classes are described as follows:

Class name	Node
Description	Declares the object of type Node which holds the location of the node of the tree
Object Variables	
degree	Integer value holds the number of children of the given node object
leftSibling	Pointer value of the left sibling of the given node object
parent	Pointer value of the parent of the given node object
rightSibling	Pointer value of the right sibling of the given node object
childNode	Pointer value of the left-most child of the given node object
keyword	String value of the keyword linked to the given node object
value	Integer value associated with the keyword that stores the frequency of occurrence
childCut	Boolean value of the childCut

Class name	keywordcounter
Description	Main class that generates the tree object and the table object which form the backbone of the program
Object Variables	
degree	Integer value holds the number of children of the given node object

Function Descriptions

Class keywordcounter

Boolean sentenceIsInsertable(String line);	
Description	Function checks the pattern of the given line and checks if matches type 1 {begins with “$” followed by a keyword and value}
Function Parameters	
Line	String contains the line from the input file to check the pattern
Return Value	
Boolean	True: if pattern is found False: if pattern is not found

Boolean sentenceIsStop(String line);	
Description	Function checks the pattern of the given line and checks if matches type 3 {stop}
Function Parameters	
Line	String contains the line from the input file to check the pattern
Return Value	
Boolean	True: if pattern is found False: if pattern is not found

void main(String[] args);	
Description	Main Function
Function Parameters	
args	String array for accepting console arguments

Class Heap

void insertToHeap(Node node_to_insert);	
Description	Accepts the node ‘node_to_insert’ and then inserts it into the heap. Update the value of maxNode
Function Parameters	
node_to_insert	Type: Node object; insert to the heap.
Return Value	
void	-

void increaseNodeValue(Node node_to_insert, int addValue);	
Description	Checks the value of the node, increase it by the given value. Check if the value is higher than the

	parent and then remove the node from its position and add it to the top level if needed.
Function Parameters	
node_to_insert	Type: Node object; insert to the heap.
addValue	Type: integer; add the value to the the node.value
Return Value	
void	-

void cutChild(Node currentNode, Node parentNode);	
Description	Check the parent childCut value and then perform cascadingCut operation if true
Function Parameters	
currentNode	Type: Node object; node needs to be removed.
parentNode	Type: Node object; parentNode updates its childCut value if it’s child is removed
Return Value	
void	-

void cascadingCut(Node node_to_insert);	
Description	CascadingCut checks the individual values of the parents whose nodes have been removed and then
Function Parameters	
node_to_insert	Type: Node object; insert to the heap.
Return Value	
void	-

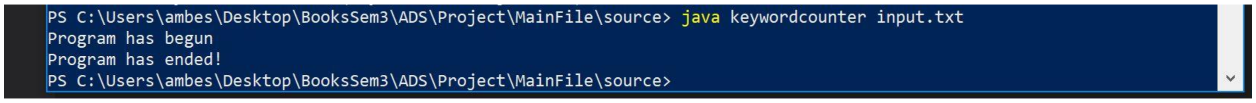
void removeMaxNode();	
Description	Removes the current node pointed by maxNode from the heap and updates the value of maxNode to the next corresponding maximum
Function Parameters	
-	-
Return Value	
void	-

void updateMaxNode();	
Description	Updates the variables of the object MaxNode to corresponding maximum of the heap
Function Parameters	
-	-
Return Value	
void	-

void pairwiseMerge();	
Description	Pairwise combines trees in the Fibonacci Heap with equal degree values
Function Parameters	
-	-
Return Value	
void	-

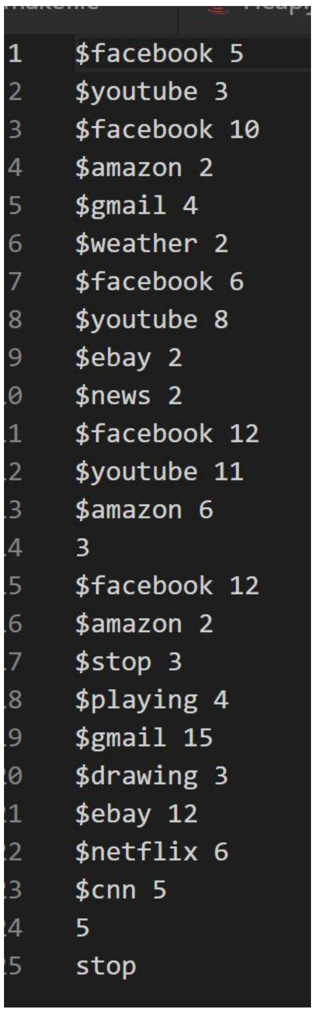
Sample Working Screenshots

Input (terminal)



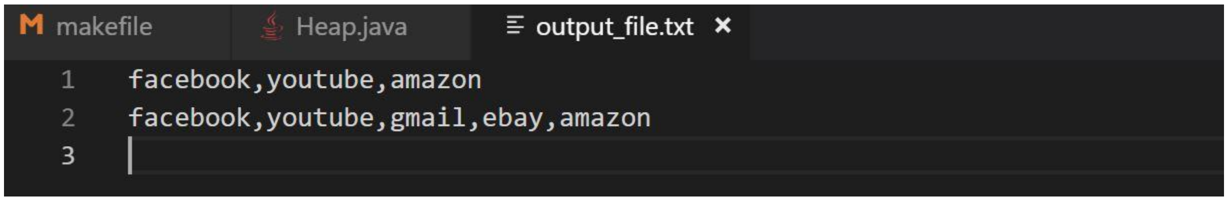
```
PS C:\Users\ambes\Desktop\BooksSem3\ADS\Project\MainFile\source> java keywordcounter input.txt
Program has begun
Program has ended!
PS C:\Users\ambes\Desktop\BooksSem3\ADS\Project\MainFile\source>
```

Input File



```
$facebook 5
$youtube 3
$facebook 10
$amazon 2
$gmail 4
$weather 2
$facebook 6
$youtube 8
$ebay 2
$news 2
$facebook 12
$youtube 11
$amazon 6
3
$facebook 12
$amazon 2
$stop 3
$playing 4
$gmail 15
$drawing 3
$ebay 12
$netflix 6
$cnn 5
5
stop
```

Output File



```
facebook,youtube,amazon
facebook,youtube,gmail,ebay,amazon

```