Nahrin Sharna

Lab 06

02/24/2019

# Initial Test Plan:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test # | Valid / Invalid Data | Description of test | Input Value | Expected Output | Actual Output | Test Pass / Fail |
| 1 | Valid | When the word contains one reoccurring character | Word1 = “zareen” | Array elements:  z a r e n  Linked List elements:  a e n r z  Occurrence: 2 for e, 1 for rest |  |  |
| 2 | Valid | When the word contains more than one reoccurring character | Word1 = “appeeal” | Array elements:  a p e l  Linked List elements:  a e l p  Occurrence: 2 for a,e,p and 1 for l |  |  |
| 3 | Invalid | When words have more 25 charachters | Word1=  ”thisislongthisislonglonger”  Word2=  “thisislongertooooomuuuuchh” | Too many characters for array  Show the list and occurrence for each word |  |  |
| 4 | Valid | Merge two array when total size is less than 25 | Word1 = “love”  Word2 = “hate” | Now the new array  after merging is:  l o v e h a t e |  |  |
| 5 | Invalid | Merge two array when total size is more than 25 | Word1 = “abcdefghijklmnop”  Word2 = “mnopqrstuvwxyz” | Arrays can not be merged because of the size |  |  |
| 6 |  |  |  |  |  |  |

# Source Code:

#include<iostream>

#include<string>

using namespace std;

struct a

{

char letter;

int occurences;

};

struct node

{

a data;

node \*next, \*prev;

node\* operator+=(node \*head) {

}

};

node\* createnode(char ch)

{

node \*n = new node;

n->data.letter = ch;

n->data.occurences = 1;

n->prev = NULL;

n->next = NULL;

return n;

}

void addnode(node \*\*head, node \*n)

{

node \*temp, \*prev;

if (\*head == NULL) //1st node is inserted

\*head = n;

else

{

temp = \*head;

prev = NULL;

while (temp != NULL && n->data.letter > temp->data.letter)

{

prev = temp;

temp = temp->next;

}

if (prev == NULL) //insert new node at 1st place

{

n->next = temp;

temp->prev = n;

\*head = n; //update head

}

else if (temp == NULL) //insert new node at end

{

prev->next = n;

n->prev = prev;

}

else

{

prev->next = n;

n->prev = prev;

n->next = temp;

temp->prev = n;

}

}

}

void printarray(a \*a, int n)

{

cout << "-----Printing the elemenets of array-----" << endl;

cout << "\ncharacter\toccurences\n";

for (int i = 0; i < n; i++)

cout << a[i].letter << "\t\t" << a[i].occurences << endl;

cout << endl;

}

void printlist(node \*p)

{

cout << "----Printing the elements of the list------" << endl;

node \*temp = p;

cout << "\ncharacter\t\toccurences\n";

while (temp != NULL)

{

cout << temp->data.letter << "\t\t" << temp->data.occurences << endl;

temp = temp->next;

}

cout << endl;

}

int main()

{

a arr1[25], arr2[25], arr3[25]; //array

node \*start1 = NULL, \*start2 = NULL, \*start3 = NULL, \*p, \*n;

string w1, w2;

char ch;

int ind1 = 0, ind2 = 0; int ind3;

int i, j;

cout << "Enter word 1: ";

cin >> w1;

cout << "Enter word 2: ";

cin >> w2;

if (w1.size() > 25)

cout << "Too many characters for array" << endl;

else {

for (i = 0; i < w1.size(); i++) //Traverse first word

{

for (j = 0; j < ind1; j++)

{

if (arr1[j].letter == w1[i]) //This letter already appeared

{

arr1[j].occurences++;

break;

}

}

if (j == ind1) //this letter appeared first time

{

arr1[ind1].letter = w1[i];

arr1[ind1].occurences = 1;

ind1++;

}

}

}

if (w2.size() > 25)

cout << "Too many characters for array" << endl;

else {

for (i = 0; i < w2.size(); i++) //Traverse second word

{

for (j = 0; j < ind2; j++)

{

if (arr2[j].letter == w2[i]) //This letter already appeared

{

arr2[j].occurences++; //increase its frequency

break;

}

}

if (j == ind2) //this letter appeared first time

{

arr2[ind2].letter = w2[i];

arr2[ind2].occurences = 1;

ind2++;

}

}

}

printarray(arr1, ind1);

printarray(arr2, ind2);

ind3 = ind1 + ind2;

if (ind3 > 25)

cout << "Arrays can not be merged because of the total size." << endl;

else {

for (i = 0; i < ind1; i++)

arr3[i].letter = arr1[i].letter;

int k;

for (i = 0, k = ind1; k < ind3 && i < ind2; i++, k++)

{

arr3[k].letter = arr2[i].letter;

}

cout << "Now the new array after merging is :\n";

for (i = 0; i < ind3; i++)

{

cout << arr3[i].letter << " ";

}

cout << endl;

}

for (i = 0; i < w1.size(); i++) //Traverse first word

{

p = start1;

while (p != NULL)

{

ch = p->data.letter;

if (ch == w1[i])

{

p->data.occurences++;

break;

}

p = p->next;

}

if (p == NULL) //this letter appeared first time

{

n = createnode(w1[i]);

addnode(&start1, n);

}

}

for (i = 0; i < w2.size(); i++) //Traverse first word

{

p = start2;

while (p != NULL)

{

if (p->data.letter == w2[i])

{

p->data.occurences++;

break;

}

p = p->next;

}

if (p == NULL) //this letter appeared first time

{

n = createnode(w2[i]);

addnode(&start2, n);

}

}

printlist(start1);

printlist(start2);

system("pause");

return 0;

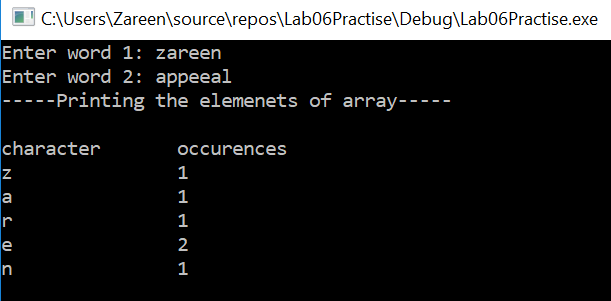
}

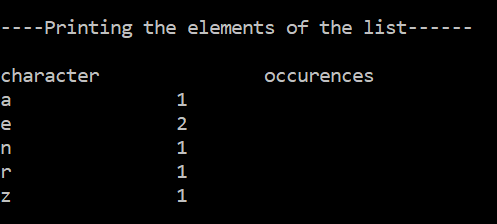
# Final Test Plan:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test # | Valid / Invalid Data | Description of test | Input Value | Expected Output | Actual Output | Test Pass / Fail |
| 1 | Valid | When the word contains one reoccurring character | Word1 = “zareen” | Array elements:  z a r e n  Linked List elements:  a e n r z  Occurrence: 2 for e, 1 for rest | Array elements:  z a r e n  Linked List elements:  a e n r z  Occurrence: 2 for e, 1 for rest | Pass |
| 2 | Valid | When the word contains more than one reoccurring character | Word1 = “appeeal” | Array elements:  a p e l  Linked List elements:  a e l p  Occurrence: 2 for a,e,p and 1 for l | Array elements:  a p e l  Linked List elements:  a e l p  Occurrence: 2 for a,e,p and 1 for l | Pass |
| 3 | Invalid | When words have more 25 charachters | Word1=  ”thisislongthisislonglonger”  Word2=  “thisislongertooooomuuuuchh” | Too many characters for array  Show the list and occurrence for each word | Liked List(w1) and occurences:  e(1)g(3) h(2) i(4) l(3) n(3) o(3) r(1) s(4) t(2)  Liked List(w1) and occurences:  c e g h i l m n o r s t u  occurences:  1 for c e g l m n r  2 for i s t  3 for h, 4 for u 6 for o | Pass |
| 4 | Valid | Merge two array when total size is less than 25 | Word1 = “love”  Word2 = “hate” | Now the new array  after merging is:  l o v e h a t e | Now the new array  after merging is:  l o v e h a t e | Pass |
| 5 | Invalid | Merge two array when total size is more than 25 | Word1 = “abcdefghijklmnop”  Word2 = “mnopqrstuvwxyz” | Arrays can not be merged because of the size | Arrays can not be merged because of the size | Pass |
| 6 |  |  |  |  |  |  |

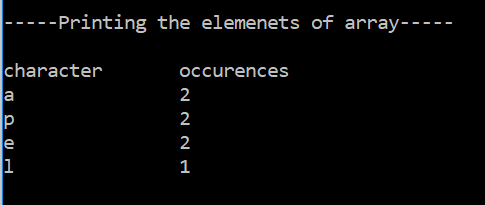
# Screenshots:

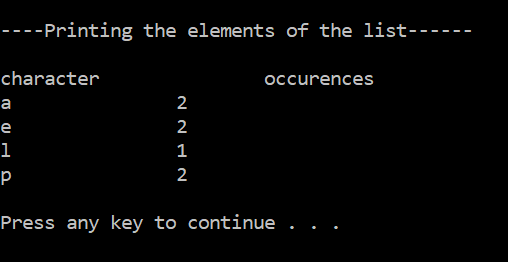
**Test Case 1:**



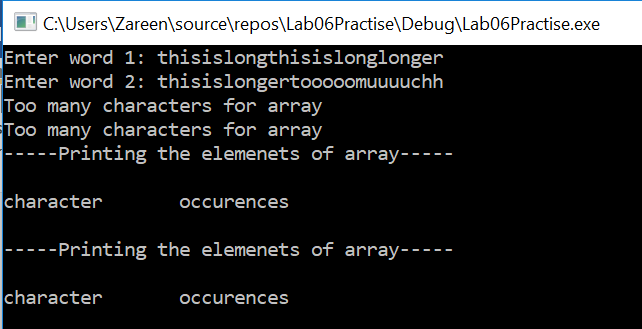


**Test Case 2:**

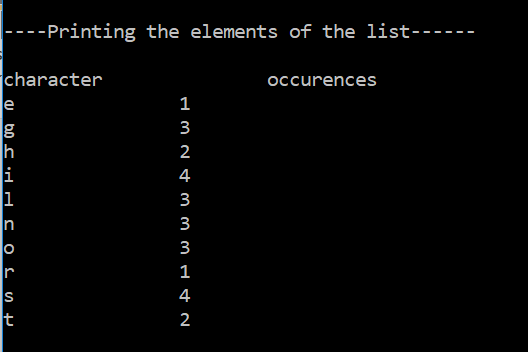




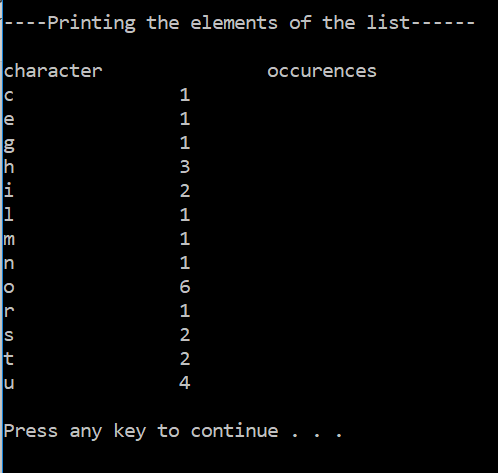
**Test Case 3:**



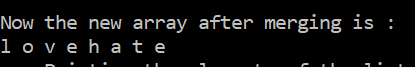
**Linked List for Word1:**



**Linked List for Word2:**



**Test Case 4:**



**Test Case 5:**

