DSPLAB

Week – 5 Assignment

Source Code:

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

struct node {

char data[30];

struct node\* child;

struct node\* sib;

}\*root=NULL;

int fmax=1; //max frequency

//insert category

int checkval=0;

void Category\_ins(struct node \*root,char\* ch,int cv){

if(cv==1){//store Word values

if(root->child != NULL){

root=root->child;

while(root->sib != NULL){

if(strcmp(root->data,ch) == 0){

if(root->child == NULL){//if word is already present then set freq to 2 when child is null

struct node\* freq=(struct node\*)malloc(sizeof(struct node));

strcpy(freq->data,"2");

checkval=1;

if(2 > fmax){// storing maxium frequency value

fmax=2;

}

freq->child=NULL;

freq->sib=NULL;

root->child=freq;

printf("\nFrequency updated\n\n");

return;

}

else{

char c1=root->child->data[0];// update the freq when it is more than 2 times repeated words

int cv=c1-'0';

cv=cv+1;

if(cv > fmax){

fmax=cv;

}

c1=cv+'0';

root->child->data[0]=c1;

printf("\nFrequency updated\n\n");

return;

}

}

root=root->sib;

}

if(strcmp(root->data,ch) == 0){// it is to check the last node

if(root->child == NULL){

struct node\* freq=(struct node\*)malloc(sizeof(struct node));

strcpy(freq->data,"2");//update frequency in the child node of the given word

checkval=1;

if(2 > fmax){

fmax=2;

}

freq->child=NULL;

freq->sib=NULL;

root->child=freq;

printf("\nFrequency updated\n\n");

return;

}

else{

char c1=root->child->data[0];// update frequency when it is more than 2 times

int cv=c1-'0';

cv=cv+1;

if(cv > fmax){

fmax=cv;

}

c1=cv+'0';

root->child->data[0]=c1;

printf("\nFrequency updated\n\n");

return;

}

}

struct node\* temp=(struct node\*)malloc(sizeof(struct node));

strcpy(temp->data,ch);

temp->child=NULL;

temp->sib=NULL;

root->sib=temp;

printf("\n word inserted\n\n");

}

else{

struct node\* temp=(struct node\*)malloc(sizeof(struct node));

strcpy(temp->data,ch);

temp->child=NULL;

temp->sib=NULL;

root->child=temp;

printf("\n word inserted\n\n");

}

}

else

{//cv==0 is to store category values

if(root->child != NULL){

root=root->child;

while(root->sib != NULL){

if(strcmp(root->data,ch) != 0){

root=root->sib;

}

else{

printf("\nCategory already exists\n\n");

return;

}

}

if(strcmp(root->data,ch) != 0){

struct node\* temp=(struct node\*)malloc(sizeof(struct node));

strcpy(temp->data,ch);

temp->child=NULL;

temp->sib=NULL;

root->sib=temp;

printf("\n Category was inserted to Tree\n\n");

}

else{

printf("\nCategory exists\n\n");

return;

}

}

else {//if tree is empty

struct node\* temp=(struct node\*)malloc(sizeof(struct node));

strcpy(temp->data,ch);

temp->child=NULL;

temp->sib=NULL;

root->child=temp;

printf("\n Category was inserted to Tree\n\n");

}

}

}

int Depth\_Count(struct node\* root){// Counts the depth of the root node

int countd=0,maxd=0;

while(root!=NULL){

struct node\* temp=root;

countd=0;

while(temp!=NULL){

countd+=1;

temp=temp->child;

}

if(maxd<countd)

{

maxd=countd;

}

root=root->sib;

}

return maxd;

}

int check\_Words=0;

void Des\_order(struct node\* root,int freq){// words in descending order

while(root!=NULL){

struct node\* check=(struct node\*)malloc(sizeof(struct node\*));

check=root->child;

while(check!=NULL){

if(check->child!=NULL){

char ch=check->child->data[0];

int value=ch-'0';

if(value==freq){

printf("%s - %d\n",check->data,freq);

check\_Words=1;

}

}

else if(check->child==NULL && freq==1){

printf("%s\n",check->data);

check\_Words=1;

}

check=check->sib;

}

root=root->sib;

}

}

void Display\_lNode(struct node\* root){//display the leaf nodes present in the tree

while(root!=NULL){

struct node\* temp=root->child;

if(temp==NULL){

printf("%s\n",root->data);

}

else{

while(temp!=NULL){

printf("%s\n",temp->data);

temp=temp->sib;

}

}

root=root->sib;

}

printf("\n");

}

int depth=0;

void Depth\_Tree(struct node\* root, char\* ch){ //function to check root present in tree and counts its depth

if(root==NULL)

return ;

if(strcmp(root->data,ch)==0){

printf("\nDepth of the Node is %d\n\n",Depth\_Count(root->child));

depth=1;

return ;

}

else{

Depth\_Tree(root->sib,ch);

Depth\_Tree(root->child,ch);

}

}

void display(struct node\* root){// display the categories with their words

//printf("\n%s\n",root->data);

root=root->child;

while(root!=NULL){

printf("\nCategory is :\n");

printf("%s\n",root->data);

struct node\* temp=root;

temp=temp->child;

printf("words are:\n");

if(temp==NULL)

printf("empty\n");

else{

while(temp!=NULL){

printf("%s",temp->data);

if(temp->child != NULL){

printf(" %s\n",temp->child->data);

}

else{

printf("\n");

}

temp=temp->sib;

}

}

root=root->sib;

}

printf("\n");

}

int DegreeCount(struct node\* root){// Counts the degree of the node

int dc=0;

while(root!=NULL){

dc+=1;

root=root->sib;

}

return dc;

}

void words\_ins(struct node \*root,char\* ch,char\* cat){//insert words into the tree

if(root->child == NULL){

printf("\n\nNo Category Exists\n\n");

return;

}

else{

root=root->child;

while(root!= NULL){

if(strcmp(root->data,cat) == 0){

break;

}

root=root->sib;

}

if(root==NULL){// when given category is not found

printf("\nNo such category found\n\n");

return;

}

else{

Category\_ins(root,ch,1);

}

}

}

int degr=0;

void Degree(struct node\* root, char\* ch){//function to search given root in tree and get its degree

if(root==NULL)

return;

if(strcmp(root->data,ch)==0){

printf("\nDegree of the Node is : %d\n\n",DegreeCount(root->child));

degr=1;

return ;

}

else{

Degree(root->child,ch);

Degree(root->sib,ch);

}

}

int main(){

struct node\* temp=(struct node\*)malloc(sizeof(struct node));

char ch[]="words";

strcpy(temp->data,ch);

temp->child=NULL;

temp->sib=NULL;

root=temp;

int choose=0;

while (choose!=8)

{

printf("\n 1.Insert Category in to Tree \n 2.Insert word to an Category\n 3.Display Tree\n 4.Display Leaf nodes\n 5.Degree of nodes\n 6.Depth of nodes \n 7.Display the words in Descending order of Frequency \n 8.Exit:\nEnter your choice : ");

if(scanf("%d", &choose)==1){

if(choose==1){

char cat[20];

printf("Enter the Category: ");

scanf("%s",&cat);

Category\_ins(root,cat,0);

}

else if(choose==2){

if(root->child == NULL)

printf("\nCategories are empty\n\n");

else{

char wd[20];

printf("Enter the word: ");

scanf("%s",&wd);

char cat[20];

printf("Enter the Category: ");

scanf("%s",&cat);

words\_ins(root,wd,cat);

}

}

else if(choose==3){

display(root);

}

else if(choose==4){

printf("\nLeaf nodes:\n");

if(root->child==NULL)

printf("%s\n",root->data);

else

Display\_lNode(root->child);

}

else if(choose==5){

printf("Enter the word/category: ");

char wc[20];

scanf("%s",&wc);

Degree(root,wc);

if(degr==0)

printf("\nNo Word found\n\n");

degr=0;

}

else if(choose==6){

printf("Enter the word/category: ");

char wc[20];

scanf("%s",&wc);

if(strcmp(wc,"words")==0 && checkval==1){

printf("\nDepth is 3\n\n");

}

else{

Depth\_Tree(root,wc);

if(depth==0)

printf("\nNo Word found\n\n");

depth=0;

}

}

else if(choose==7){

if(root->child != NULL){

printf("\n");

int b=fmax;

while(b>0){

Des\_order(root->child,b);

b--;

}

if(check\_Words==0)

printf("No words exist\n");

printf("\n");

}

else{

printf("\nNo words Exist\n\n");

}

}

else if(choose==8)

break;

else

printf("\nEnter valid choice\n\n");

}

else{

printf("\nEnter Only Numbers\n");

exit(0);

}

}

}

Output::







