Header

#include<cstdlib>

using namespace std;

template <class T>

class linklist{

public:

linklist();

~linklist();

linklist \*create\_list(T&);

void insert\_beginning(T&);

void insert\_last(T&);

void insert\_position(T&, long);

void update\_node(T&, long);

void display();

void delete\_node(T&);

void reverse\_list();

void search\_node(T&);

void sort\_list();

void delete\_position(T&);

bool isempty();

T \*access\_list();

long accs\_totnumlist();

T array, rev, \*list;

linklist \*temp, \*next, \*prev, \*start, \*p, \*s;

long count;

};

template<class T>

linklist<T>::linklist(){

list=0;

start=NULL;

}

template<class T>

linklist<T>::~linklist(){

delete[] list;

}

template<class T>

linklist<T>\* linklist<T>::create\_list(T& newitem){

temp=new linklist;

if(temp==NULL){

cout<<"Memory not allocated";

return 0;

}

else{

temp->array=newitem;

temp->next=NULL;

temp->prev=NULL;

return temp;

}

}

template<class T>

void linklist<T>::insert\_beginning(T& newitem){

temp=create\_list(newitem);

if(start==NULL){

start=temp;

start->next=NULL;

start->prev=NULL;

return;

}

else{

s=start;

start=temp;

start->next=s;

s->prev=start;

start->prev=NULL;

return;

}

}

template<class T>

void linklist<T>::insert\_last(T& newitem){

temp=create\_list(newitem);

if(start==NULL){

start=temp;

start->next=NULL;

start->prev=NULL;

return;

}

else{

s=start;

while(s->next!=NULL){

s=s->next;

}

s->next=temp;

temp->next=NULL;

temp->prev=NULL;

return;

}

}

template<class T>

void linklist<T>::delete\_node(T& newitem){

if(start==NULL){

cout<<"Empty";

return;

}

while(start->array==newitem){

start=start->next;

}

temp=start;

while(temp->next!=NULL){

if(temp->next->array==newitem){

s=temp->next;

temp->next=temp->next->next;

delete s;

}

else{

temp=temp->next;

}

}

}

template<class T>

void linklist<T>::delete\_position(T& pos){

long a;

if(start==NULL){

// cout<<"Empty";

return;

}

if(pos==1){

p=start;

start=p->next;

delete p;

return;

}

s=start;

a=1;

while(s->next!=NULL){

if(pos-1==a){

p=s->next;

s->next=p->next;

delete p;

return;

}

a++;

s=s->next;

}

// cout<<"invalid pos";

}

template<class T>

void linklist<T>::display(){

if(start==NULL){

cout<<"Empty";

return;

}

else{

temp=start;

while(temp!=NULL){

cout<<temp->array<<"->";

temp=temp->next;

}

cout<<"NULL";

}

}

template<class T>

void linklist<T>::insert\_position(T& newitem, long pos){

long a;

temp=create\_list(newitem);

if(pos==1){

s=start;

start=temp;

start->next=s;

return;

}

else{

s=start;

a=1;

while(s->next!=NULL){

if(pos-1==a){

p=s->next;

s->next=temp;

temp->next=p;

return;

}

a++;

s=s->next;

}

cout<<"invalid position";

}

}

template<class T>

void linklist<T>::update\_node(T& newitem, long pos){

long a;

if(start==NULL){

cout<<"empty";

return;

}

else{

temp=start;

a=1;

while(temp!=NULL){

if(pos==a){

temp->array=newitem;

return;

}

a++;

temp=temp->next;

}

cout<<"invalid position";

}

}

template<class T>

void linklist<T>::search\_node(T& newitem){

long a;

if(start==NULL){

cout<<"Empty";

return;

}

else{

temp=start;

a=1;

cout<<"Item "<<newitem<<" found at position : ";

while(temp!=NULL){

if(temp->array==newitem){

cout<<a<<" ";

}

temp=temp->next;

a++;

}

}

}

template<class T>

void linklist<T>::reverse\_list(){

temp=start;

s=start->next;

p=NULL;

while(s!=NULL){

temp->next=p;

p=temp;

temp=s;

s=s->next;

}

start=temp;

temp->next=p;

}

/\* reverse algo

1 2 3 n

t s p

1 2 n

2 n 1

2 3 n

3 n 2

3 2

\*/

template<class T>

void linklist<T>::sort\_list(){

if(start==NULL){

cout<<"Empty";

return;

}

else{

s=start;

while(s!=NULL){

temp=start;

while(temp->next!=NULL){

if(temp->array>temp->next->array){

rev=temp->array;

temp->array=temp->next->array;

temp->next->array=rev;

}

temp=temp->next;

}

s=s->next;

}

}

}

template<class T>

bool linklist<T>::isempty(){

return(start==NULL);

}

template<class T>

T\* linklist<T>::access\_list(){

count=accs\_totnumlist();

list = new T[count];

long a=0;

temp=start;

while(temp!=NULL){

list[a]=temp->array;

// cout<<list[a]<<" ";

a++;

temp=temp->next;

}

return list;

}

template<class T>

long linklist<T>::accs\_totnumlist(){

temp=start;

long mycount=0;

while(temp!=NULL){

temp=temp->next;

mycount++;

}

return mycount;

}

Body

#‎include‬ <cstdlib>

#include <iostream>

#include "llist.h"

using namespace std;

void option();

long comparestring\_ana(char[], char[]);

int main(int argc, char \*argv[])

{

char words[100], \*mychar[20], \*tokenptr;

long a,b,c,mycond;

string anag;

char str1[30], str2[30];

linklist<string> mylink;

cout<<"Enter list of words here: ";

gets(words);

tokenptr=strtok(words," ");

a=0;

while(tokenptr!=NULL){

mychar[a]=tokenptr;

tokenptr=strtok(NULL, " ");

a++;

}

b=0;

while(b<a){

c=b+1;

while(c<a){

mycond=0;

strcpy(str1, mychar[b]);

strcpy(str2, mychar[c]);

mycond=comparestring\_ana(str1,str2);

if(mycond==1){

anag=str1;

mylink.insert\_last(anag);

anag=str2;

mylink.insert\_last(anag);

}

c++;

}

b++;

}

mylink.display();

cout<<endl<<endl;

system("PAUSE");

return EXIT\_SUCCESS;

}

long comparestring\_ana(char mychar1[], char mychar2[]){

int i,j,k,l,mycond,count1,count2;

if(strlen(mychar1)==strlen(mychar2)){

j=0;

mycond=0;

while(j<strlen(mychar1) && mycond!=1){

i=0;

count1=1;

while(i<strlen(mychar1)){

if(i!=j && mychar1[i]==mychar1[j]){

count1++;

}

i++;

}

k=0;

count2=0;

while(k<strlen(mychar2)){

if(mychar2[k]==mychar1[j]){

count2++;

}

k++;

}

if(count1!=count2){

mycond=1;

}

j++;

}

if(mycond==1){

return 0;

}

else

return 1;

}

else{

return 0;

}

}