#include<iostream>

#include<string>

#include<fstream>

#include<conio.h>

#include<iomanip>

#include<Windows.h>

#include<unistd.h>

using namespace std;

typedef struct Node

{

string data;

struct Node\* prev;

struct Node\* next;

}word;

word \*head;

word \*cursor;

typedef struct Stack

{

string data;

struct Stack \*link;

}stack;

stack \*top;

typedef struct STACK

{

word \*Head;

word \*Cursor;

struct STACK \*link;

}UNDO;

UNDO \*HEAD;

UNDO \*REDO\_HEAD;

word\* insert(string data,word \*head)

{

word\* temp=new word;

temp->data=data;

if(head==NULL)

{

temp->prev=NULL;

temp->next=NULL;

head=temp;

cursor=temp;

}

else if(cursor==NULL)

{

temp->next=head;

head->prev=temp;

head=temp;

cursor=temp;

}

else

{

temp->next=cursor->next;

temp->prev=cursor;

if(cursor->next!=NULL)

cursor->next->prev=temp;

cursor->next=temp;

cursor=temp;

}

return head;

}

void Delete()

{

word \*ptr=cursor;

if(ptr==head)

{

head=NULL;

}

else if(ptr->next==NULL)

{

ptr->prev->next=NULL;

cursor=ptr->prev;

}

else

{

ptr->prev->next=ptr->next;

ptr->next->prev=ptr->prev;

cursor=ptr->prev;

}

delete ptr;

}

void print()

{

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

CONSOLE\_SCREEN\_BUFFER\_INFO consoleInfo;

WORD savedAttributes;

GetConsoleScreenBufferInfo(hConsole, &consoleInfo);

savedAttributes = consoleInfo.wAttributes;

word \*ptr=head;

while(ptr!=NULL)

{

if(cursor==ptr)

{

SetConsoleTextAttribute(hConsole, FOREGROUND\_BLUE);

}

else

{

SetConsoleTextAttribute(hConsole, savedAttributes);

}

cout<<ptr->data<<" ";

ptr=ptr->next;

}

SetConsoleTextAttribute(hConsole, savedAttributes);

cout<<"\n";

}

word\* find(string key);

void move\_cursor()

{

int pos;

int space;

int i;

string key;

cout<<"where to move the cursor?\n1-left\n2-right\n3-to a word\n4-to last\n5-tobeggining\nchoice: ";

cin>>pos;

word \*ptr;

switch(pos)

{

case 1:

cout<<"how many spaces to move: ";

cin>>space;

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

{

cursor=cursor->prev;

}

break;

case 2:

cout<<"how many spaces to move: ";

cin>>space;

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

{

if(cursor->next!=NULL)

cursor=cursor->next;

}

break;

case 3:

cout<<"enter the word: ";

cin>>key;

ptr=find(key);

cursor=ptr;

break;

case 4:

while(cursor->next!=NULL)

{

cursor=cursor->next;

}

break;

case 5:

cursor=head;

break;

default:

cout<<"invalid choice\n";

}

}

word\* find(string key)

{

word \*ptr=head;

char ch;

while(ptr!=NULL)

{

if(ptr->data==key)

{

cursor=ptr;

print();

cout<<"this occurence(y-yes)? ";

cin>>ch;

if(ch=='y')

return ptr;

else

{

ptr=ptr->next;

system("cls");

}

}

else

{

ptr=ptr->next;

}

}

cout<<"word not found\n";

return NULL;

}

void Search(string key)

{

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

CONSOLE\_SCREEN\_BUFFER\_INFO consoleInfo;

WORD savedAttributes;

GetConsoleScreenBufferInfo(hConsole, &consoleInfo);

savedAttributes = consoleInfo.wAttributes;

word \*ptr=head;

while(ptr!=NULL)

{

if(ptr->data==key)

{

SetConsoleTextAttribute(hConsole, FOREGROUND\_GREEN);

cout<<ptr->data<<" ";

SetConsoleTextAttribute(hConsole, savedAttributes);

}

else

{

cout<<ptr->data<<" ";

}

ptr=ptr->next;

}

cout<<"\n";

}

void writetext()

{

string data;

cout<<"ENTER the text(type # to stop): \n";

do

{

cin>>data;

if(data=="#")

break;

head=insert(data,head);

}while(1);

}

void deletetext()

{

string from;

string to;

cout<<"enter from which word to delete: ";

cin>>from;

word \*ptr=find(from);

if(ptr!=NULL)

{

cout<<"enter upto which word to delete: ";

cin>>to;

word \*ptr1=find(to);

if(ptr1!=NULL)

{

cursor=ptr1;

while(cursor!=ptr)

{

Delete();

}

Delete();

}

else

cout<<"the word not in text\n";

}

else

{

cout<<"the word not in text\n";

}

}

void pushstack(string data)

{

stack \*temp=new stack;

temp->data=data;

if(top==NULL)

top=temp;

else

{

temp->link=top;

top=temp;

}

}

string popstack()

{

string data;

if(top==NULL)

return "#";

else

{

data=top->data;

stack \*temp=top;

top=top->link;

delete temp;

return data;

}

}

void empty\_stack()

{

if(top!=NULL)

{

stack \*ptr;

while(top!=NULL)

{

ptr=top;

delete ptr;

top=top->link;

}

}

}

void cut()

{

empty\_stack();

string from;

string to;

cout<<"enter from which word to cut: ";

cin>>from;

word \*ptr=find(from);

if(ptr!=NULL)

{

cout<<"enter upto which word to cut: ";

cin>>to;

word \*ptr1=find(to);

if(ptr1!=NULL)

{

cursor=ptr1;

while(cursor!=ptr)

{

pushstack(cursor->data);

Delete();

}

pushstack(ptr->data);

Delete();

}

else

cout<<"the word not in text\n";

}

else

{

cout<<"the word not in text\n";

}

}

void Copy()

{

top=NULL;

empty\_stack();

string from;

string to;

cout<<"enter from which word to copy: ";

cin>>from;

word \*ptr=find(from);

if(ptr!=NULL)

{

cout<<"enter upto which word to copy: ";

cin>>to;

word \*ptr1=find(to);

if(ptr1!=NULL)

{

cursor=ptr1;

while(cursor!=ptr)

{

pushstack(cursor->data);

cursor=cursor->prev;

}

pushstack(ptr->data);

}

else

cout<<"the word not in text\n";

}

else

{

cout<<"the word not in text\n";

}

}

void paste()

{

string data;

data=popstack();

while(data!="#")

{

head=insert(data,head);

data=popstack();

}

}

void Replace()

{

int ch;

string data;

string data1;

cout<<"Enter the word which you want to replace: ";

cin>>data;

cout<<"enter the word to replace the existing word: ";

cin>>data1;

do{

cout<<"want to replace \n1.all\n or \n2.particluar occurences: ";

cin>>ch;

if(ch==1||ch==2)

break;

else

cout<<"invalid choice\ntry again!\n";

}while(1);

if(ch==1)

{

system("cls");

Search(data);

word\* ptr=head;

while(ptr!=NULL)

{

if(data==ptr->data)

{

ptr->data=data1;

}

ptr=ptr->next;

}

}

else

{

word \*ptr=head;

char ch;

while(ptr!=NULL)

{

if(ptr->data==data)

{

cursor=ptr;

system("cls");

print();

cout<<"this occurence(y=yes)? ";

cin>>ch;

if(ch=='y')

{

ptr->data=data1;

}

else

{

ptr=ptr->next;

system("cls");

}

}

else

{

ptr=ptr->next;

}

}

}

system("cls");

cout<<"REPLACED\n";

Search(data1);

while(getchar()!='\n');

while(getchar()!='\n');

}

UNDO\* saveoperation(UNDO \*HEAD1)

{

if(HEAD==NULL)

{

word \*c=cursor;

UNDO \*Temp=(UNDO\*)malloc(sizeof(UNDO));

word \*ptr1=head;

word \*newhead=NULL;

while(ptr1!=NULL)

{

newhead=insert(ptr1->data,newhead);

ptr1=ptr1->next;

}

Temp->Head=newhead;

Temp->Cursor=cursor;

cursor=c;

HEAD1=Temp;

}

else

{

word\* c=cursor;

UNDO \*Temp=(UNDO\*)malloc(sizeof(UNDO));

word\* ptr1=head;

word\* newhead=NULL;

while(ptr1!=NULL)

{

newhead=insert(ptr1->data,newhead);

ptr1=ptr1->next;

}

Temp->Head=newhead;

Temp->Cursor=cursor;

cursor=c;

Temp->link=HEAD1;

HEAD1=Temp;

}

return HEAD1;

}

void undo()

{

if(HEAD!=NULL)

{

UNDO \*tem=HEAD;

HEAD=HEAD->link;

REDO\_HEAD=saveoperation(REDO\_HEAD);

cursor=tem->Cursor;

head=tem->Head;

}

}

void redo()

{

if(REDO\_HEAD!=NULL)

{

UNDO \*tem=REDO\_HEAD;

REDO\_HEAD=REDO\_HEAD->link;

HEAD=saveoperation(HEAD);

cursor=tem->Cursor;

head=tem->Head;

}

}

int main()

{

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

CONSOLE\_SCREEN\_BUFFER\_INFO consoleInfo;

WORD savedAttributes;

GetConsoleScreenBufferInfo(hConsole, &consoleInfo);

savedAttributes = consoleInfo.wAttributes;

SetConsoleTextAttribute(hConsole, FOREGROUND\_BLUE);

sleep(3);

cout<<"\n\n\n\n\n\n\n";

cout<<"----------------------------------------------------------------------------------------------------------------------------------------------------------------------------";

cout<<"\n\t\t\t\t\t\t\t";

SetConsoleTextAttribute(hConsole, savedAttributes);

sleep(1);

cout<<"WELCOME";

sleep(1);

cout<<" TO";

sleep(1);

cout<<" TEXT ";

sleep(1);

cout<<"EDITOR\n";

sleep(1);

cout<<"\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\tDONE BY\n\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\tNIVITHASRI A\n\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\tTANAZ I\n";

SetConsoleTextAttribute(hConsole, FOREGROUND\_BLUE);

cout<<"----------------------------------------------------------------------------------------------------------------------------------------------------------------------------";

while(getchar()!='\n');

system("cls");

SetConsoleTextAttribute(hConsole, savedAttributes);

head=NULL;

cursor=NULL;

int c;

int choice;

char ch;

string data;

char filename[100];

cout<<"\n\n\n\n\n\n\n";

cout << "\t||=======================================================================||" << endl;

cout << "\t||-----------------------------------------------------------------------||" << endl;

cout << "\t|| Press [1] CREATE NEW FILE ||" << endl;

cout << "\t||-----------------------------------------------------------------------||" << endl;

cout << "\t|| Press [2] EDIT EXISTING FILE ||" << endl;

cout << "\t||=======================================================================||" << endl;

do{cout<<"\nENTER CHOICE: ";

cin>>choice;

if(choice==1||choice==2)

break;

else

cout<<"invalid choice\ntry again!\n";

}while(1);

if(choice==1)

{

cout<<"enter file name: ";

cin>>filename;

ofstream f;

f.open(filename);

f.close();

system("cls");

writetext();

}

else

{

cout<<"enter filename to open: ";

cin>>filename;

ifstream f1;

string data;

f1.open(filename);

while(f1)

{

f1>>data;

head=insert(data,head);

}

f1.close();

}

do

{

system("cls");

print();

cout<<"\n1.write\t2.delete a word\t3.move cursor\t4.delete a particular text\t5.search\t6.cut\t7.copy\t8.paste\t9.replace\t10.undo\t11.redo\n0.EXIT\nenter choice: ";

cin>>c;

switch(c)

{

case 1:

HEAD=saveoperation(HEAD);

writetext();

break;

case 2:

HEAD=saveoperation(HEAD);

Delete();

break;

case 3:

HEAD=saveoperation(HEAD);

move\_cursor();

break;

case 4:

HEAD=saveoperation(HEAD);

deletetext();

break;

case 5:

cout<<"enter word to search: ";

cin>>data;

Search(data);

while(getchar()!='\n');

while(getchar()!='\n');

break;

case 6:

HEAD=saveoperation(HEAD);

cut();

break;

case 7:

Copy();

break;

case 8:

HEAD=saveoperation(HEAD);

paste();

break;

case 9:

HEAD=saveoperation(HEAD);

Replace();

break;

case 10:

undo();

break;

case 11:

redo();

break;

case 0:

break;

default:

cout<<"invalid choice. TRY AGAIN\n";

}

if(c==0)

break;

}while(1);

ofstream f2;

f2.open(filename);

string words;

word \*ptr=head;

char dec;

cout<<"SAVE CHANGES MADE TO THE FILE(y-yes)? ";

cin>>dec;

if(dec=='y')

{

while(ptr!=NULL)

{

words=ptr->data;

f2<<words;

f2<<' ';

ptr=ptr->next;

}

}

f2.close();

}