**DSA FISAC-1**

**Name: Drishaan Prasad  
Registration Number: 210905282  
Roll Number: 47  
Branch: CSE  
Section: A**

1.

#include<stdio.h>

#include<math.h>

#include<stdlib.h>

struct node{

int val;

struct node\*next;

};

struct node\*head1 = NULL;

struct node\*tail1 = NULL;

struct node\*head2 = NULL;

struct node\*tail2 = NULL;

void addNode1(int val){

if(!head1){

head1 = (struct node \*)malloc(sizeof(struct node));

tail1 = head1;

head1->val = val;

head1->next = NULL;

} else {

tail1->next = (struct node\*)malloc(sizeof(struct node));

tail1 = tail1->next;

tail1->val = val;

tail1->next = NULL;

}

}

void addNode2(int val){

if(!head2){

head2 = (struct node \*)malloc(sizeof(struct node));

tail2 = head2;

head2->val = val;

head2->next = NULL;

} else {

tail2->next = (struct node\*)malloc(sizeof(struct node));

tail2 = tail2->next;

tail2->val = val;

tail2->next = NULL;

}

}

void printList(struct node\*temp){

while(temp != NULL){

printf("%d ",temp->val);

temp = temp->next;

}

printf("\n");

}

struct node\* merge(struct node\*h1,struct node\*h2){

if(!h1){

return h2;

}

if(!h2){

return h1;

}

struct node\*h = NULL;

struct node\*t = NULL;

if(h1->val < h2->val){

h = h1;

h1 = h1->next;

t = h;

} else {

h = h2;

h2 = h2->next;

t = h;

}

while(h1!=NULL && h2!=NULL){

if(h1->val < h2->val){

t->next = h1;

t = t->next;

h1 = h1->next;

} else {

t->next = h2;

t = t->next;

h2 = h2->next;

}

}

while(h1 != NULL){

t->next = h1;

t = h1;

h1 = h1->next;

}

while(h2 != NULL){

t->next = h2;

t = h2;

h2 = h2->next;

}

return h;

}

int main(){

printf("LIST 1: ");

addNode1(2);

addNode1(4);

addNode1(6);

addNode1(8);

printList(head1);

printf("LIST 2: ");

addNode2(3);

addNode2(5);

addNode2(7);

addNode2(9);

printList(head2);

struct node\* res = merge(head1,head2);

printf("After merging:\nLIST 1: ");

printList(res);

}  
  
Text

Description automatically generated

2.

#include<stdio.h>

#include<math.h>

#include<stdlib.h>

struct node{

int val;

struct node\*next;

};

struct node\*head1 = NULL;

struct node\*tail1 = NULL;

void addNode1(int val){

if(!head1){

head1 = (struct node \*)malloc(sizeof(struct node));

tail1 = head1;

head1->val = val;

head1->next = NULL;

} else {

tail1->next = (struct node\*)malloc(sizeof(struct node));

tail1 = tail1->next;

tail1->val = val;

tail1->next = NULL;

}

}

void printList(struct node\*temp){

while(temp != NULL){

printf("%d ",temp->val);

temp = temp->next;

}

printf("\n");

}

int getSize(struct node\*h){

int count = 0;

while(h != NULL){

count++;

h = h->next;

}

return count;

}

struct node\*rightRotate(int n,struct node\*head){

if(getSize(head) == n || n == 0){

return head;

}

int size = getSize(head);

struct node\*temp = head;

struct node\*prev = NULL;

struct node\*newhead = head;

for(int i=0;i<(size-n);i++){

prev = temp;

temp = temp -> next;

}

while(newhead->next != NULL){

newhead = newhead->next;

}

newhead->next = head;

prev->next = NULL;

return temp;

}

struct node\*leftRotate(int n,struct node\*head){

if(getSize(head) == n || n == 0){

return head;

}

struct node\*temp = head;

struct node\*prev = NULL;

struct node\*t = head;

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

prev = temp;

temp = temp->next;

}

while(t->next != NULL){

t = t->next;

}

prev->next = NULL;

t->next = head;

return temp;

}

int main(){

int n;

printf("enter size of list: ");

scanf("%d",&n);

int val;

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

printf("enter the node %d: ",i+1);

scanf("%d",&val);

addNode1(val);

}

struct node\*copy = NULL;

struct node\*tcopy = NULL;

struct node\*temp = head1;

while(temp != NULL){

if(!copy){

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

copy->val = temp->val;

tcopy = copy;

copy->next = NULL;

} else {

tcopy->next = (struct node\*)malloc(sizeof(struct node));

tcopy->next->val = temp->val;

tcopy = tcopy->next;

tcopy->next = NULL;

}

temp = temp->next;

}

printf("original list: ");

printList(head1);

printf("copied list: ");

printList(copy);

printf("left rotate: ");

struct node\*res2 = leftRotate(2,copy);

printList(res2);

printf("right rotate: ");

struct node\*res1 = rightRotate(2,head1);

printList(res1);

}

Text

Description automatically generated

3.

#include<stdio.h>

#include<math.h>

#include<stdlib.h>

struct node{

int val;

struct node\*next;

};

struct node\*head1 = NULL;

struct node\*tail1 = NULL;

struct node\*head2 = NULL;

struct node\*tail2 = NULL;

void addNode1(int val){

if(!head1){

head1 = (struct node \*)malloc(sizeof(struct node));

tail1 = head1;

head1->val = val;

head1->next = NULL;

} else {

tail1->next = (struct node\*)malloc(sizeof(struct node));

tail1 = tail1->next;

tail1->val = val;

tail1->next = NULL;

}

}

void addNode2(int val){

if(!head2){

head2 = (struct node \*)malloc(sizeof(struct node));

tail2 = head2;

head2->val = val;

head2->next = NULL;

} else {

tail2->next = (struct node\*)malloc(sizeof(struct node));

tail2 = tail2->next;

tail2->val = val;

tail2->next = NULL;

}

}

void printList(struct node\*temp){

while(temp != NULL){

printf("%d ",temp->val);

temp = temp->next;

}

printf("\n");

}

struct node\*fib(int n){

head1 = (struct node\*)malloc(sizeof(struct node));

head1->val = 0;

struct node\*t = head1;

struct node\*prev = NULL;

if(n == 1){

return head1;

}

prev = t;

t->next = (struct node\*)malloc(sizeof(struct node));

t = t->next;

t->val = 1;

if(n == 2){

return head1;

}

for(int i = 0 ; i<(n-2); i++){

t->next = (struct node\*)malloc(sizeof(struct node));

// t = t->next;

t->next->val = t->val + prev->val;

prev = t;

t = t->next;

}

return head1;

}

int main(){

int n;

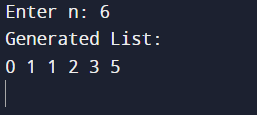
printf("Enter n: ");

scanf("%d",&n);

struct node\*res = fib(n);

printf("Generated List:\n");

printList(res);

}  
  


4.

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

struct node{

char name[50];

int roll\_no;

int marks;

struct node\*next;

};

struct node\*head = NULL;

struct node\*tail = NULL;

void addNode(char name[],int rn,int marks){

if(!head){

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

tail = head;

strcpy(head->name,name);

head->roll\_no = rn;

head->marks = marks;

tail->next = NULL;

} else {

tail->next = (struct node\*)malloc(sizeof(struct node));

tail = tail->next;

strcpy(tail->name,name);

tail->roll\_no = rn;

tail->marks = marks;

tail->next = NULL;

}

}

void sortList(struct node\*head){

struct node\*i;

struct node\*j;

for(i = head; i != NULL; i = i->next){

for(j = i->next; j != NULL; j = j->next){

if(j->marks > i->marks){

struct node temp;

strcpy(temp.name,i->name);

temp.roll\_no = i->roll\_no;

temp.marks = i->marks;

strcpy(i->name,j->name);

i->roll\_no = j->roll\_no;

i->marks = j->marks;

strcpy(j->name,temp.name);

j->roll\_no = temp.roll\_no;

j->marks = temp.marks;

}

}

}

}

void printList(struct node\*temp){

while(temp != NULL){

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

printf("%d ",temp->roll\_no);

printf("%d ",temp->marks);

temp = temp->next;

}

printf("\n");

}

int main(){

addNode("Arun",12,45);

addNode("Ram",6,89);

addNode("Raju",1,60);

printList(head);

printf("Sorted list based on marks: \n");

sortList(head);

printList(head);

printf("Second highest: \n");

printf("%s %d %d",head->next->name,head->next->roll\_no,head->next->marks);

}

Text

Description automatically generated

5.

**insert\_functions.h**

#include<stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node\* next;

};

struct node\* insertFirst(struct node\* head, int item)

{

}

struct node\* insertEnd(struct node\* head, int item)

{

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

ptr->data=item;

ptr->next=NULL;

struct node\* p = head;

if(p==NULL)

return ptr;

else

{

while(p->next!=NULL)

{

p=p->next;

}

p->next=ptr;

}

return head;

}

struct node\* insertBetween(struct node\* head, int item)

{

}

void display(struct node \*ptr)

{

while (ptr!=NULL)

{

printf("%d ",ptr->data);

ptr = ptr->next;

}

}

**q5.c**

#include<stdio.h>

#include<stdlib.h>

#include<math.h>

#include "insert\_functions.h"

void main()

{

int n,i,item;

struct node\* list;

int choice;

printf("1. Create a singly linked list of N numbers and display\n");

printf("2. Delete every occurrence of the numbers in the linked list which are equal to the square of the given key value\n");

printf("3. Display the updated list after the deletion of elements\n");

printf("4. Exit\n");

do

{

printf("\nEnter your choice: ");

scanf("%d",&choice);

if(choice==1)

{

printf("Enter number of nodes to create: ");

scanf("%d",&n);

list = (struct node\*)calloc(n,sizeof(struct node));

list=NULL;

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

{

printf("Enter data of node %d: ",i+1);

scanf("%d",&item);

list=insertEnd(list,item);

}

printf("\nData in list before deletion: \n");

display(list);

printf("\n");

}

else if(choice==2)

{

struct node\* p = list;

struct node\* q;

printf("Enter element value of key to be considered for deletion: ");

scanf("%d",&item);

item\*=item;

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

{

if((p->data)==item)

{

if(p==list)

{

p=p->next;

list=p;

}

else if(p->next==NULL)

{

q->next=NULL;

}

else

{

q->next=p->next;

p=p->next;

}

}

else

{

q=p;

p=p->next;

}

}

}

else if(choice==3)

{

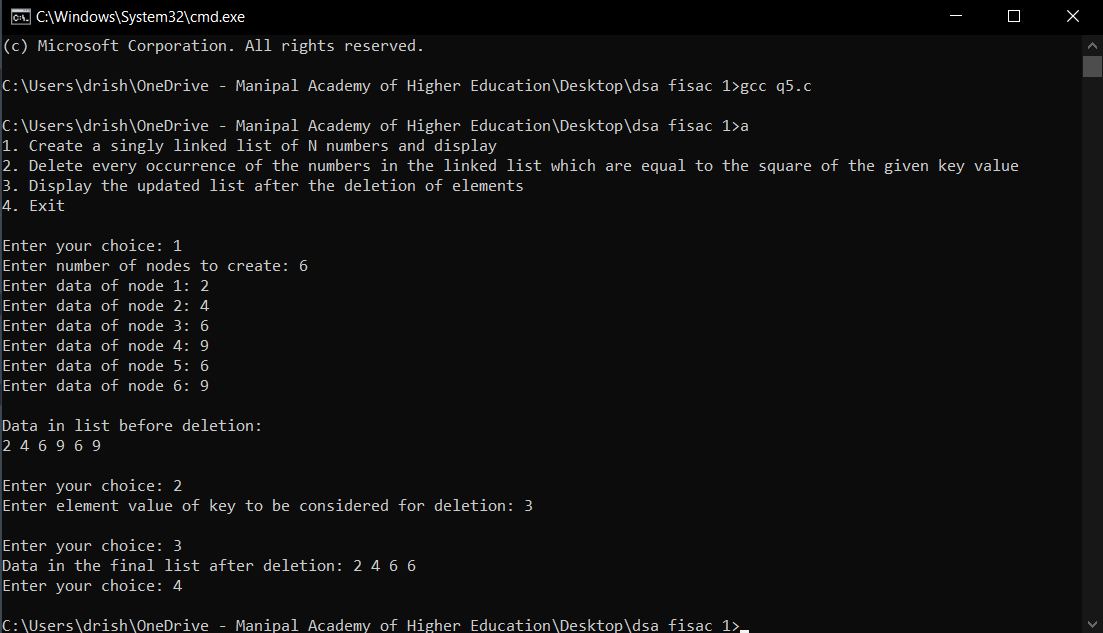
printf("Data in the final list after deletion: ");

display(list);

}

} while(choice!=4);

}



6.

#include<stdio.h>

#include<math.h>

#include<stdlib.h>

struct node{

int val;

struct node\*next;

};

struct node\*head1 = NULL;

struct node\*tail1 = NULL;

void addNode1(int val){

if(!head1){

head1 = (struct node \*)malloc(sizeof(struct node));

tail1 = head1;

head1->val = val;

head1->next = NULL;

} else {

tail1->next = (struct node\*)malloc(sizeof(struct node));

tail1 = tail1->next;

tail1->val = val;

tail1->next = NULL;

}

}

void printList(struct node\*temp){

while(temp != NULL){

printf("%d ",temp->val);

temp = temp->next;

}

printf("\n");

}

int main(){

addNode1(1);

addNode1(2);

addNode1(3);

addNode1(4);

addNode1(5);

addNode1(6);

printf("Sample nodes of the original list: \n");

printList(head1);

struct node\*temp = head1;

// printf("%d hi",temp->val);

struct node\*eve = NULL;

struct node\*odd = NULL;

struct node\*etail = NULL;

struct node\*otail = NULL;

while(temp){

if(temp->val % 2 == 0){

if(!eve){

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

etail = eve;

eve->val = temp->val \* temp->val \* temp->val;

etail->next = NULL;

} else {

etail->next = (struct node \*)malloc(sizeof(struct node));

etail = etail->next;

etail->val = temp->val \* temp->val \* temp->val;

etail->next = NULL;

}

} else {

if(!odd){

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

otail = odd;

odd->val = temp->val \* temp->val \* temp->val;

otail->next = NULL;

} else {

otail->next = (struct node \*)malloc(sizeof(struct node));

otail = otail->next;

otail->val = temp->val \* temp->val \* temp->val;

otail->next = NULL;

}

}

temp = temp->next;

}

printf("Displaying the elements of LIST-1 followed by LIST-2:\nLIST1: ");

printList(eve);

printf("LIST2: ");

printList(odd);

}

Text

Description automatically generated

7.

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

struct node

{

char data[50];

struct node\* next;

};

struct node\* insert(struct node\* head, char str[])

{

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

strcpy((ptr->data),str);

ptr->next=NULL;

struct node\* p = head;

if(p==NULL)

return ptr;

else

{

while(p->next!=NULL)

{

p=p->next;

}

p->next=ptr;

}

return head;

}

void display(struct node \*ptr)

{

while (ptr!=NULL)

{

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

ptr = ptr->next;

}

}

int isPalin(char str[])

{

int i,j,flag=1;

for(i=0,j=strlen(str)-1;i<=j;i++,j--)

{

if(str[i]!=str[j])

{

flag=0;

break;

}

}

return flag;

}

void main()

{

int n,i;

char s[50],temp[50];

printf("enter size of list: ");

scanf("%d",&n);

struct node\* list=NULL;

printf("enter strings: \n");

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

{

scanf(" %s",&s);

list=insert(list,s);

}

printf("original list: ");

display(list);

printf("\n");

struct node\* PSLIST=NULL;

struct node\* NPSLIST=NULL;

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

{

strcpy(temp,list->data);

if(isPalin(temp))

{

PSLIST=insert(PSLIST,temp);

}

else

{

NPSLIST=insert(NPSLIST,temp);

}

list=list->next;

}

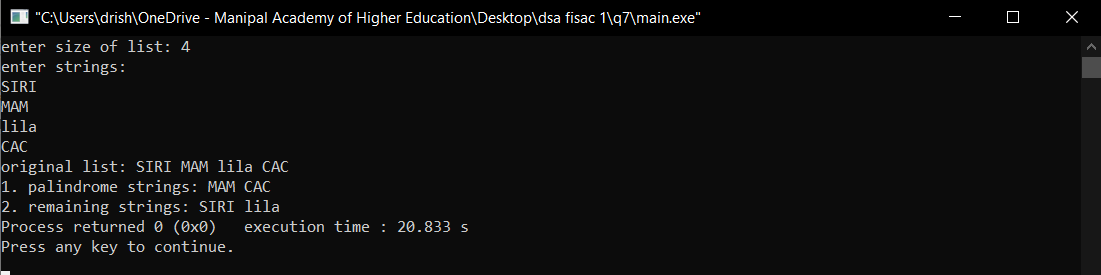
printf("1. palindrome strings: ");

display(PSLIST);

printf("\n2. remaining strings: ");

display(NPSLIST);

}



8.

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <stdbool.h>

struct node {

int data;

int key;

struct node \*next;

};

struct node \*head = NULL;

struct node \*current = NULL;

struct node \*l1 = NULL;

struct node \*l2 = NULL;

//display the list

void printList() {

struct node \*ptr = head;

printf("\n[ ");

//start from the beginning

while(ptr != NULL) {

printf("(%d) ",ptr->data);

ptr = ptr->next;

}

printf("]");

}

//insert link at the first location

void insertFirst(int key, int data) {

//create a link

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

link->key = key;

link->data = data;

//point it to old first node

link->next = head;

//point first to new first node

head = link;

}

void reverse(struct node\*\* head\_ref) {

struct node\* prev = NULL;

struct node\* current = \*head\_ref;

struct node\* next;

while (current != NULL) {

next = current->next;

current->next = prev;

prev = current;

current = next;

}

\*head\_ref = prev;

}

bool isEmpty() {

return head == NULL;

}

void main() {

printf("List : ");

insertFirst(1,8);

insertFirst(2,4);

insertFirst(3,6);

insertFirst(4,2);

insertFirst(5,1);

printList();

printf("\n");

struct node \*start1 = head;

struct node \*start2 = head->next;

int key1=1,key2=1;

while(start1!=NULL)

{

struct node \*f1 = (struct node \*)malloc(sizeof(struct node));

f1->key = key1;

key1++;

f1->data=start1->data;

f1->next=l1;

l1=f1;

if(start1->next==NULL)

{

break;

}

start1=start1->next->next;

}

while(start2!=NULL)

{

struct node \*f2 = (struct node \*)malloc(sizeof(struct node));

f2->key = key2;

key2++;

f2->data=start2->data;

f2->next=l2;

l2=f2;

if(start2->next==NULL)

{

break;

}

start2=start2->next->next;

}

reverse(&l1);

reverse(&l2);

printf("List 1:\n");

while(l1!=NULL)

{

printf("%d->",l1->data);

l1=l1->next;

}

printf("null\n");

printf("List 2:\n");

while(l2!=NULL)

{

printf("%d->",l2->data);

l2=l2->next;

}

printf("null\n");

}

Text

Description automatically generated

9.

**delete\_functions.h**

#include<stdio.h>

#include<stdlib.h>

struct node

{

char data;

struct node\* next;

};

struct node\* deleteFirst(struct node\* head)

{

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

ptr=head;

head=ptr->next;

free(ptr);

return head;

}

struct node\* deleteEnd(struct node\* head)

{

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

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

ptr=head;

while(ptr->next!=NULL)

{

p=ptr;

ptr=ptr->next;

}

free(ptr);

p->next=NULL;

return head;

}

struct node\* deleteBetween(struct node\* head, int index)

{

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

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

ptr=head;

int i;

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

{

p=ptr;

ptr=ptr->next;

}

p->next=ptr->next;

free(ptr);

return head;

}

**q9.c**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include "delete\_functions.h"

int isVowel(char ch)

{

if(ch=='a' || ch=='A' || ch=='e' || ch=='E' || ch=='i' || ch=='I' || ch=='o' || ch=='O' || ch=='u' || ch=='U')

return 1;

else

return 0;

}

struct node\* create(char name[])

{

int i;

struct node\* list=NULL;

for(i=0;i<strlen(name);i++)

{

char item=name[i];

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

ptr->data=item;

ptr->next=NULL;

struct node\* p = list;

if(p==NULL)

list=ptr;

else

{

while(p->next!=NULL)

{

p=p->next;

}

p->next=ptr;

}

}

return list;

}

void display(struct node \*ptr)

{

while (ptr!=NULL)

{

printf("%c ",ptr->data);

ptr = ptr->next;

}

}

void allVowels(struct node\* head)

{

while (isVowel(head->data)) {

head = head->next;

}

struct node\* p = head;

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

struct node\* list = head;

while(p!=NULL)

{

char ch=p->data;

if(isVowel(ch))

{

q->next=p->next;

} else {

q=p;

}

p=p->next;

}

printf("\nmodified name: ");

display(list);

}

void main()

{

char name[50];

int i;

printf("enter a name: ");

gets(name);

struct node\* list=NULL;

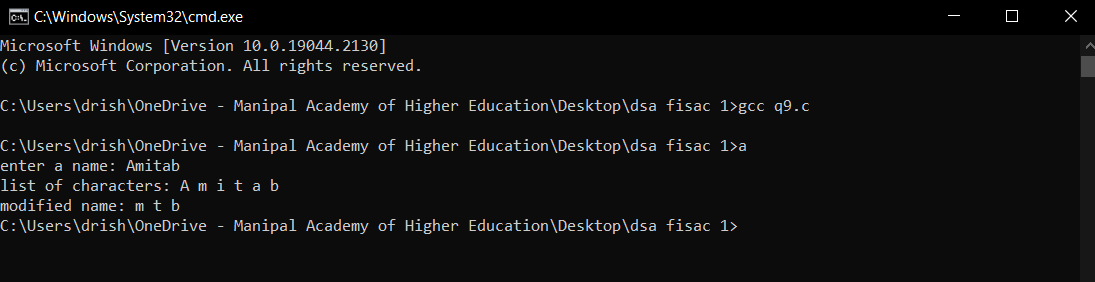
list=create(name);

printf("list of characters: ");

display(list);

allVowels(list);

}



10.

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

struct node

{

char data[50];

int count;

struct node\* next;

};

struct node\* create(struct node\* head,char name[])

{

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

strcpy(ptr->data,name);

ptr->next=NULL;

struct node\* p = head;

if(p==NULL){

return ptr;

}

else

{

while(p->next!=NULL)

{

p=p->next;

}

p->next=ptr;

}

return head;

}

void search(struct node\* head)

{

struct node\* p = head;

struct node \*q, \*temp;

int count;

while(p != NULL){

temp = p;

q = p->next;

count=1;

while(q != NULL) {

if(!strcmp(p->data, q->data)) {

temp->next = q->next;

count++;

}

else {

temp = q;

}

q = q->next;

}

printf("\n%s %d ", p->data, count);

p = p->next;

}

}

void display(struct node \*ptr)

{

while (ptr!=NULL)

{

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

ptr = ptr->next;

}

}

int main()

{

char name[50];

int i,n;

printf("enter number of names: ");

scanf("%d",&n);

struct node\* list=NULL;

printf("enter the names: \n");

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

{

scanf("%s",&name);

list=create(list,name);

}

printf("list of first names: ");

display(list);

search(list);

printf("\nmodified list: ");

display(list);

return 0;

}

