11、

#include<stdio.h>

struct intNode

{

int val;

struct intNode \*next;

};

struct intNode \*findmin(struct intNode \*a,struct intNode \*\*pre)

{

struct intNode \*temp, \*p,\*q,\*mpre;

for(temp=p=a,mpre=q=NULL;p;q=p,p=p->next)

if(temp->val>p->val)

{

mpre=q;

temp=p;

}

\*pre=mpre;

return temp;

}

int del(struct intNode \*\*a)

{

struct intNode \*p,\*q;

p=findmin(\*a,&q);

if(!p)

return 0;

if(!q)

\*a=p->next;

else

q->next=p->next;

free(p);

return 1;

}

12、

#include<stdio.h>

#include<stdlib.h>

struct Node {

int z;

int m;

struct Node \*next;

};

typedef struct Node Dota;

int cmp(Dota \*a, Dota \*b) {

return a->z \* b->m - b->z \* a->m; //×??o????

}

void output(Dota \*head) {

while(head) {

printf(" %d/%d ", head->z, head->m);

head = head->next;

}

}

Dota \*insert(Dota \*head, Dota\* p) {

if(head == NULL) return p;

Dota\* t= head;

if(t->next != NULL) {

while(cmp(t->next, p) < 0){

t = t->next;

}

if(cmp(t->next, p) == 0) {

free(p);

return head;

}

}

p->next = t->next;

t->next = p;

//output(head);

return head;

}

int main() {

int n = 10;

Dota \*head = NULL;

Dota\* p = (Dota \*)malloc(sizeof(Dota));

p->z = 0;

p->m = 1;

p->next = 0;

head = insert(head, p);

//output(head);

p = (Dota \*)malloc(sizeof(Dota));

p->z = 1;

p->m = 1;

p->next = 0;

head = insert(head, p);

//output(head);

int i,j;

for(i = 1; i < n; i++) {

for(j = n; j > i; j--) {

p = (Dota \*)malloc(sizeof(Dota));

p->z = i;

p->m = j;

p->next = 0;

head = insert(head, p);

}

}

output(head);

return 0;

}

13、

#include<stdio.h>

#include<stdlib.h>

struct Node {

char c;

int v;

struct Node \*next;

};

typedef struct Node Dota;

Dota\* newlist(int len) {

if(len == 0) return NULL;

Dota\* head = (Dota \*)malloc(sizeof(Dota));

head->v = 0;

head->next = head;

Dota\* last = head;

while(--len) {

Dota \* p= (Dota \*)malloc(sizeof(Dota));

p->v = 0;

p->next = last->next;

last->next = p;

last = last->next;

}

return head;

}

Dota\* to\_next(Dota\* p) {

p = p->next;

while(p->v){

p = p->next;

}

return p;

}

Dota\* encode(char s[], int len, int key) {

Dota \* head = newlist(len);

Dota \* empty = head;

int i = 0;

for(i = 0; i < len; i++) {

int t = key;

while(--t){

empty = to\_next(empty);

}

empty->c = s[i];

empty->v = 1;

if(i < len - 1){

empty = to\_next(empty);

}

}

return head;

}

int main() {

char s[] = "CN dota, best dota.";

int key = 4;

Dota\* t = encode(s, sizeof(s), key);

}

14、

void plus(struct intNode \*&a, struct intNode \*b)

{

struct intNode \*pa = a->next, \*sa = a, \*pb = b->next;

while (pb != NULL && pa->ival > pb->ival)

{

struct intNode \*temp = pb;

sa->next = temp;

pb = pb->next;

temp->next = pa;

sa = temp;

}

while (pb != NULL)

{

if (pa->ival < pb->ival)

{

struct intNode \*temp = pb;

pb = pb->next;

temp->next = pa->next;

pa->next = temp;

pa = temp->next;

sa = temp;

}

else if (pa->ival == pb->ival)

pb = pb->next;

else

{

struct intNode \*temp = pb;

pb = pb->next;

temp->next = pa;

sa->next = temp;

pa = pa->next;

sa = temp;

}

}

return;

}

void minus(struct intNode \*&a, struct intNode \*b)

{

struct intNode \*pa = a->next, \*sa = a, \*pb = b->next;

while (pb != NULL&&pa != NULL)

{

if (pb->ival == pa->ival)

{

sa->next = pa->next;

free(pa);

pa = sa->next;

pb = pb->next;

}

else if (pb->ival > pa->ival)

{

pa = pa->next;

sa = sa->next;

}

else

pb = pb->next;

}

return;

}

void bothhave(struct intNode \*&a, struct intNode \*b)

{

struct intNode \*pa = a->next, \*sa = a, \*pb = b->next;

while (pb != NULL && pa != NULL)

{

if (pb->ival == pa->ival)

{

pa = pa->next;

sa = sa->next;

}

else

{

sa->next = pa->next;

free(pa);

pa = sa->next;

}

pb = pb->next;

}

return;

}

15、

struct intNode

{

int ival;

struct intNode \*next;

};

struct intNode \*plus(struct intNode \*a, struct intNode \*b)

{

struct intNode \*head=NULL, \*tail=NULL, \*p, \*pa=a, \*pb=b;

while(pa!=NULL||pb!=NULL)

{

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

if(pa==NULL||(pb!=NULL&&pb->ival>pa->ival))

{

p->ival=pb->ival;

pb=pb->next;

}

else

{

p->ival=pa->ival;

pa=pa->next;

}

p->next=NULL;

if(tail!=NULL&&p->ival==tail->ival)

continue;

if(!head)

head=tail=p;

else

tail=tail->next=p;

}

return head;

}

struct intNode \*minus(struct intNode \*a, struct intNode \*b)

{

struct intNode \*head=NULL, \*tail=NULL, \*p, \*pa=a, \*pb=b;

while(1)

{

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

if(pa==NULL)

break;

else if(pb==NULL||pa->ival<pb->ival)

{

p->ival=pa->ival;

pa=pa->next;

}

else if(pa->ival==pb->ival)

{

pa=pa->next;

pb=pb->next;

continue;

}

else

{

pa=pa->next;

continue;

}

p->next=NULL;

if(!head)

head=tail=p;

else

tail=tail->next=p;

}

return head;

}

struct intNode \*bothhave(struct intNode \*a, struct intNode \*b)

{

struct intNode \*head=NULL, \*tail=NULL, \*p, \*pa=a, \*pb=b;

while(pa&&pb)

{

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

if(pa->ival==pb->ival)

{

p->ival=pa->ival;

pa=pa->next;

pb=pb->next;

}

else if(pa->ival<pb->ival)

{

pa=pa->next;

continue;

}

else

{

pb=pb->next;

continue;

}

p->next=NULL;

if(!head)

head=tail=p;

else

tail=tail->next=p;

}

return head;

}