6、

#include<stdio.h>

#include<time.h>

#include<stdlib.h>

#include<string.h>

struct Card

{

char flower[10];

char value[4];

int ival;

}cards[52];//生成牌的步骤在main里

char values[13][3] = { "A","2","3","4","5","6","7","8","9","10","J","Q","K" };

char names[4][10] = { "heitao","fangpian","hongtao","meihua" };

void xipai(struct Card \*cards, struct Card \*to)

{

struct Card temp[52];

srand(time(NULL));

for (int j = 0; j < 52; j++)

temp[j] = cards[j];

for (int count = 0; count < 52; count++)

{

int select = (rand()) % (52 - count);

to[count] = temp[select];

for (int j = select; j < 51 - count - 1; j++)

temp[j] = temp[j + 1];

}

}

void discard(struct Card all[52], struct Card \*p1, struct Card \*p2, struct Card \*p3, struct Card \*p4)

{

for (int i = 0; i < 13; i++)

{

p1[i] = all[4 \* i];

p2[i] = all[4 \* i + 1];

p3[i] = all[4 \* i + 2];

p4[i] = all[4 \* i + 3];

}

}

int compare(struct Card a, struct Card b)//自变量为结构数组中的元素

{

if (a.ival > b.ival)

return 1;

else if (a.ival == b.ival)

return 0;

else

return -1;

}

int main()

{

for (int i = 0; i < 13; i++)

{

for (int j=0,k=0;k<4;k++)

{

j = i \* 4 + k;

strcpy(cards[j].flower,names[k]);

strcpy(cards[j].value, values[i]);

cards[j].ival = i;

}

}

struct Card to[52];

xipai(cards, to);

struct Card p1[13], p2[13], p3[13], p4[13];

discard(to, p1, p2, p3, p4);

return 0;

}

7、

#include <stdlib.h>

#include <stdio.h>

struct Node {

int x;

struct Node \*next;

};

typedef struct Node Dota;

Dota \*cat\_one(Dota \*&a) {

if(!a) return NULL;

Dota \*t = a;

a = a->next;

return t;

}

Dota \*get\_next(Dota \*&a, Dota \*&b) {

if(!a) return cat\_one(b);

if(!b) return cat\_one(a);

if(a->x > b->x) {

return cat\_one(b);

} else {

return cat\_one(a);

}

}

Dota \*merge(Dota \*a, Dota \*b) {

if(!a) return b;

if(!b) return a;

Dota \*head = get\_next(a, b);

Dota \*tmp = head;

do {

tmp->next = get\_next(a, b);

} while(a || b);

return head;

}

8、

Dota \*copy1(Dota \*a) {

if(!a) return NULL;

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

head->x = a->x;

head->next = NULL;

a = a->next;

Dota \*tmp = head;

while(a) {

tmp->next = malloc(sizeof(dota));

tmp = tmp->next;

tmp->x = a->x;

tmp->next = NULL;

a = a->next;

}

return head;

}

Dota \*copy2(Dota \*a) {

if(!a) return 0;

Dota \*head = NULL;

while(a) {

Dota \*t = malloc(sizeof(dota));

t->x = a->x;

t->next = head;

head = t;

a = a->next;

}

return head;

}

Dota \*insert\_order(Dota \*head, int x) {

Dota \* t = malloc(sizeof(dota));

t->x = x;

t->next = 0;

if(!head || head->x < x) {

t->next = head;

head = t;

return head;

}

Dota \*p = head;

while(p->next->x < x) {

p = p->next;

}

t->next = p->next;

p->next = t;

return head;

}

Dota \*copy3(Dota \*a) {

if(!a) return 0;

Dota \*head = 0;

while(a) {

head = insert\_order(head, a->x);

a = a->next;

}

return head;

}

9、

struct intNode

{

int i;

struct intNode \*next;

};

int firstmin(struct intNode \*a, struct intNode \*b, struct intNode \*c)

{

while (a&&b&&c)

{

if (a->i == b->i&&a->i == c->i)

return a->i;

if (a->i < b -> i)

{

if (a->i < c->i)

a = a->next;

else

c = c->next;

}

else

{

if (b->i < c->i)

b = b->next;

else

c = c->next;

}

}

return -9999;

}

10、

struct Node {

int x;

struct Node \*next\_input;

struct Node \*next\_order;

};

typedef struct Node Dota2;

Dota2 \*insert\_order(Dota2 \*head, Dota2 \*t) {

if(!head || head->x < t->x) {

t->next\_order = head;

head = t;

return head;

}

Dota \*p = head;

while(p->next\_order->x < t->x) {

p = p->next\_order;

}

t->next\_order = p->next\_order;

p->next\_order = t;

return head;

}

void insert(Dota2 \*&headi, Dota2\*& headc, int x) {

Dota2 \* t = malloc(sizeof(Dota2));

t->x = x;

t->next\_input = 0;

t->next\_order = 0;

if(!headi) {

headc = headi = t;

}

Dota2 \* p = head;

while(p->next\_input){

p = p->next\_input;

}

p->next\_input = t;

headc = insert\_order(headc, t);

}