# 6.5

typedef struct *celltype*

{

    records data;

    celltype \*next;

} *List*;

void InsertSort(int *arr*[])

{

    List \*head;

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

    head->data = arr[1];

    for (int i = 1; i < n; i++)

    {

        if (arr[i] < head->data)

        {

            List \*y;

            y = (\*List)malloc(sizeof(List));

            y->data = arr[i];

            y->next = head;

            head = y;

        }

        List \*x = head;

        for (; x->next != NULL; x = x->next)

            if (arr[i] <= x->next->data) break;

        List \*y;

        y = (\*List)malloc(sizeof(List));

        y->data = arr[i];

        y->next = x->next;

        x->next = y;

    }

    for (List \*x = head; x != NULL; x = x->next) printf("%d ", x->data);

}

void SelectSort(int *arr*[])

{

    List \*head, \*last;

    for (int i = 1; i <= n; i++)

    {

        int minx = 0x7fffffff, x;

        for (int j = i; j <= n; j++)

            if (arr[j] < minx) minx = arr[j], x = j;

        List \*y;

        y = (\*List)malloc(sizeof(List));

        y->data = minx;

        y->next = NULL;

        if (i == 1)

            head = y;

        else

        {

            last->next = y;

        }

        last = y;

        int t = arr[x];

        arr[x] = arr[i];

        arr[i] = t;

    }

    for (List \*x = head; x != NULL; x = x->next) printf("%d ", x->data);

}

# 6.8

输入：

a 3.00 0

b 4.00 8

c 3.00 3

d 3.00 5

e 3.00 7

输出：

a 3.00 0

e 3.00 7

c 3.00 3

d 3.00 5

b 4.00 8

**可以发现：用浮点数为关键字时e 3.00 7这一行相对于其他关键字相同的数据的位置改变，说明快速排序是不稳定的。**

# 6.10

int l, r, top = 1;

    sta[top].l = 1;

    sta[top].r = n;

    while (top > 0)

    {

        l = sta[top].l;

        r = sta[top].r;

        top--;

        base = a[l];

        int h = l, t = r;

        while (h < t)

        {

            while (h < t && a[t].grade >= base.grade) t--;

            while (h < t && a[h].grade <= base.grade) h++;

            if (h < t)

            {

                tmp = a[h];

                a[h] = a[t];

                a[t] = tmp;

            }

        }

        a[l] = a[h];

        a[h] = base;

        if (l <= h - 1)

        {

            top++;

            sta[top].l = l;

            sta[top].r = h - 1;

        }

        if (h + 1 <= r)

        {

            top++;

            sta[top].l = h + 1;

            sta[top].r = r;

        }

    }