```c
// Interval Scheduling

#include<stdio.h>
#include<stdlib.h>
static int k=0;
struct data* dt=NULL;
struct is{
    int start;
    int finish;
};

struct data{
    int v;
    struct is* array;
};

struct data* create_data(int v){

    struct data* temp=(struct data*)malloc(sizeof(struct data));
    temp->v=v;
    temp->array=(struct is*)malloc(v*sizeof(struct is));
    return temp;
}

void add_data(int start,int finish){
     dt->array[k].start=start;
     dt->array[k++].finish=finish;
}
void swap(int* a, int* b)
{
    int t = *a;
    *a = *b;
    *b = t;
}

int partition ( int low, int high)
{
    int pivot = dt->array[high].finish;
    int i = (low - 1);

    for (int j = low; j <= high- 1; j++)
    {
        if (dt->array[j].finish <= pivot)
        {
            i++;
            swap(&dt->array[i].finish,&dt->array[j].finish);
             swap(&dt->array[i].start,&dt->array[j].start);
        }
    }
    swap(&dt->array[i+1].finish, &dt->array[high].finish);
    swap(&dt->array[i+1].start,&dt->array[high].start);

    return (i + 1);
}
```

```
55:
56: void quickSort(struct is* arr,int low, int high)
57: {
58:     if (low < high)
59:     {
60:         int pi = partition( low, high);
61:         quickSort(arr, low, pi - 1);
62:         quickSort(arr, pi + 1, high);
63:     }
64: }
65:
66: void interval_scheduling(){
67:
68:     quickSort(dt->array,0,dt->v);
69:     int i=0;
70:     printf("%d--%d",dt->array[i].start,dt->array[i].finish);
71:     for(int j=1;j<dt->v;++j){
72:         if(dt->array[j].start>=dt->array[i].finish)
73:         {  printf("\n%d--%d",dt->array[j].start,dt->array[j].finish);
74:           i=j;
75:         }
76:     }
77:
78: }
79:
80:
81:
82: int main(){
83:     dt=create_data(6);
84:     add_data(5,9);
85:     add_data(1,2);
86:     add_data(3,4);
87:     add_data(0,6);
88:     add_data(5,7);
89:     add_data(8,9);
90:     interval_scheduling();
91:     return 0;
92:
93: }
94:
95:
96:
```