#include<iostream>

#include<graphics.h>

#include<stdlib.h>

#include<string.h>

#include <string>

#include<conio.h>

#include<algorithm>

using namespace std;

struct work {

int xh;

int begin;

int end;

};

bool compare(work x, work y);

int main()

{

int n, r, i, j, count = 1, m, maxend = 0, preg, prek;

char ss[55];

work s[105];

scanf\_s("%d", &n);

m = n;

while (m--)

{

scanf\_s("%d", &r);

scanf\_s("%d %d", &s[r].begin, &s[r].end);

s[r].xh = r;

if (s[r].end > maxend) maxend = s[r].end;

}

initgraph(1000, 480, SHOWCONSOLE | NOCLOSE);

setbkcolor(WHITE);

cleardevice();

setcolor(BLACK);

line(30, 450, 30, 10);

line(30, 450, 900, 450);

line(25, 15, 30, 10);

line(35, 15, 30, 10);

line(895, 445, 900, 450);

line(895, 455, 900, 450);

preg = 800 / maxend;

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

{

setlinestyle(PS\_DASHDOT);

line(30 + i \* preg, 450, 30 + i \* preg, 30);

snprintf(ss, 10, "%d",i);

outtextxy(30 + i \* preg, 460, ss);

}

outtextxy(30 + maxend \* preg + 30, 460, "时间");

Sleep(150);

prek = 350 / (2 \* (n + 1));

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

{

setlinestyle(PS\_SOLID);

rectangle(s[i].begin \* preg + 30, 60 + ((2 \* i - 1) - 1)\*prek, s[i].end\*preg + 30, 60 + (2 \* i - 1)\*prek);

Sleep(355);

}

int kk;

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

{

setcolor(WHITE);

setlinestyle(PS\_SOLID);

rectangle(s[i].begin\*preg + 30, 60 + ((2 \* i - 1) - 1)\*prek, s[i].end\*preg + 30, 60 + (2 \* i - 1)\*prek);

for (kk = 1; kk <= maxend; kk++)

{

setcolor(BLACK);

setlinestyle(PS\_DASHDOT);

line(30 + kk \* preg, 450, 30 + kk \* preg, 30);

}

Sleep(120);

}

setcolor(BLACK);

line(30, 450, 30, 30);

line(30, 450, 900, 450);

sort(s + 1, s + n + 1, compare);

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

{

setlinestyle(PS\_SOLID);

rectangle(s[i].begin \* preg + 30, 60 + ((2 \* i - 1) - 1)\*prek, s[i].end\*preg + 30, 60 + (2 \* i - 1)\*prek);

snprintf(ss, 10, "%d", s[i].xh);

outtextxy((s[i].begin\*preg + 30 + s[i].end\*preg + 30 ) / 2 , (60 + ((2 \* i - 1) - 1)\*prek + 60 + (2 \* i - 1)\*prek) / 2 + 10, ss);

Sleep(350);

}

j = 1;

printf("活动为：");

printf("%d ", s[1].xh);

setfillcolor(LIGHTGREEN);

fillrectangle(s[1].begin\*preg + 30, 60 + ((2 \* 1 - 1) - 1)\*prek, s[1].end\*preg + 30, 60 + (2 \* 1 - 1)\*prek);

Sleep(155);

setfillcolor(LIGHTBLUE);

fillrectangle(s[1].begin\*preg + 30, 60 + ((2 \* (n + 1) - 1) - 1)\*prek, s[1].end\*preg + 30, 60 + (2 \* (n + 1) - 1)\*prek);

snprintf(ss, 10, "%d", s[1].xh);

outtextxy((s[1].begin\*preg + 30 + s[1].end\*preg + 30) / 2, 70 + (2 \* (n + 1) - 1)\*prek, ss);

Sleep(155);

for (i = 2; i <= n; i++)

{

if (s[i].begin >= s[j].end)

{

setfillcolor(LIGHTGREEN);

fillrectangle(s[i].begin\*preg + 30, 60 + ((2 \* i - 1) - 1)\*prek, s[i].end\*preg + 30, 60 + (2 \* i - 1)\*prek);

Sleep(155);

setfillcolor(LIGHTBLUE);

fillrectangle(s[i].begin\*preg + 30, 60 + ((2 \* (n + 1) - 1) - 1)\*prek, s[i].end\*preg + 30, 60 + (2 \* (n + 1) - 1)\*prek);

snprintf(ss, 10, "%d", s[i].xh);

outtextxy((s[i].begin\*preg + 30 + s[i].end\*preg + 30) / 2, 70 + (2 \* (n + 1) - 1)\*prek, ss);

Sleep(155);

printf("%d ", s[i].xh);

j = i;

count++;

}

}

printf("\n活动数为：%d\n", count);

\_getch();

closegraph();

return 0;

}

bool compare(work x, work y)

{

if (x.end != y.end)

return x.end < y.end;

return x.begin < y.begin;

}



