1. **页码:8　　行数:5，6　　印次:1**

**修改“**≤**”为“**<=**”, 修改“**≥**”为“**>=**”：**

|  |  |  |  |
| --- | --- | --- | --- |
| 运 算 符 | 运 算 | 范 例 | 结 果 |
| == | 相等于 | 4==3 | 0 |
| != | 不等于 | 4!=3 | 1 |
| < | 小于 | 4<3 | 0 |
| > | 大于 | 4>3 | 1 |
| <= | 小于或等于 | 4<=3 | 0 |
| >= | 大于或等于 | 4>=3 | 1 |

1. **页码:8　　行数:-6　　印次:1**

“a=n++;”**改为**“a=++n;”

1. **页码:19　　行数:25　　印次:1**

训练1-21代码中“int n,count;”**改为**“int n,count=0;”

1. **页码:22　　行数:-2　　印次:1**

训练1-27代码中“int sum;”**改为**“int sum=0;”

1. **页码:22　　行数:-5　　印次:1**

**修改训练1-30代码以下红色部分：**

char \*half(string s){

int n=s.length()/2;

char \*str=new char[n+1];

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

str[i]=s[i];

str[n]='\0';

return str;

}

1. **页码:63　　行数:17,18　　印次:1**

**i和j交换位置：**

（2）从右向左扫描，找小于或等于pivot的数r[*j*]。

（3）从左向右扫描，找大于pivot的数r[*i*]。

1. **页码:76　　行数:17　　印次:1**

**图中第三行修改“1,-1”为“1,-2”，修改“1,1”为“1,2”：**



1. **页码:173　　行数:19　　印次:1**

**“1”改为 “ pid ”：**

（7）获取当前指令，即第pid个程序的第p[pid]单元，cur=prg[pid][p[pid]]，执行该指令，然后 p[pid]++，指向第pid个程序的下一条指令。如果时间周期未用完，则继续执行该程序

1. **页码:174　　行数:-2　　印次:1**

**“1”改为 “ 2 ”：**

（8）获取当前指令，第2个程序的第p[2]（p[2]=0）单元，当前运行指令为cur=“a=3”。val[0]=3，p[2]++，此时p[2]=1，指向第2个程序的下一条指令。时间周期已用完，将2号程序

1. **页码:175　　行数:3　　印次:1**

**“1”改为 “ 3 ”：**

（10）获取当前指令，第3个程序的第p[3]（p[3]=0）单元，当前运行指令为cur=“b = 5”。val[1]=5，p[3]++，此时p[3]=1，指向第3个程序的下一条指令。时间周期已用完，将3号程序

1. **页码:176　　行数:3，7　　印次:1**

**第3行：“1”改为 “ 2 ”：**

（14）获取当前指令，第2个程序的第p[2]（p[2]=1）单元，当前运行指令为cur=“print a”，该指令是第2个程序中的输出，val[0]=3，因此输出2: 3，p[2]++，此时p[2]=2，指向第2个程

**第7行：“1”改为 “ 3 ”：**

（16）获取当前指令，第3个程序的第p[3]（p[3]=1）单元，当前运行指令为cur=“a=17”。val[0]=17，p[3]++，此时p[3]=2，指向第3个程序的下一条指令。时间周期已用完，将3号程

1. **页码:177　　行数:-2　　印次:1**

**“1”改为 “ 3 ”：**

（22）获取当前指令，第3个程序的第p[3]（p[3]=2）单元，当前运行指令为cur=“print a”，该指令是第3个程序中的输出，val[0]=17，因此输出3:17。p[3]++，此时p[3]=3，指向第3个

1. **页码:178　　行数:7　　印次:1**

**“1”改为 “ 3 ”：**

（26）获取当前指令，第3个程序的第p[3]（p[3]=3）单元，当前运行指令为cur=“print b”，该指令是第3个程序中的输出，val[1]=9，因此输出3:9。p[3]++，此时p[3]=4，指向第3个程

1. **页码:180　　行数:3，-2　　印次:1**

**第3行：“1”改为 “ 2 ”：**

（34）获取当前指令，第2个程序的第p[2]（p[2]=2）单元，当前运行指令为cur=“lock”，locked=true，p[2]++，此时p[2]=3，指向第2个程序的下一条指令。时间周期已用完，将2号程

**倒数第2行：“1”改为 “ 2 ”：**

（38）获取当前指令，第2个程序的第p[2]（p[2]=3）单元，当前运行指令为cur=“b=8”，令val[1]=8。p[2]++，此时p[2]=4，指向第2个程序的下一条指令。时间周期已用完，将2号程

1. **页码:181　　行数:6　　印次:1**

**“1”改为 “ 2 ”：**

（42）获取当前指令，第2个程序的第p[2]（p[2]=4）单元，当前运行指令为cur=“print b”，该指令是第2个程序中的输出，val[1]=8，因此输出2: 8。p[2]++，此时p[2]=5，指向第2个程序的下一条指令。时间周期已用完，将2号程序加入就绪队列的队尾。

**readQ队列内容改为2：**

 

1. **页码:182　　行数:3,6,10　　印次:1**

**第3行：“1”改为 “ 2 ”：**

（44）获取当前指令，第2个程序的第p[2]（p[2]=5）单元，当前运行指令为cur=“unlock”，令locked=false；当阻止队列不为空时，将阻止队列的第1个加入就绪队列队首。p[2]++，此时 p[2]=6，指向第2个程序的下一条指令。时间周期已用完，将2号程序加入就绪队列的

**第6行：“1”改为 “ 3”：**

（46）获取当前指令，第3个程序的第p[3]（p[3]=4）单元，当前运行指令为cur=“lock”，locked=true，p[3]++，此时p[3]=5，指向第3个程序的下一条指令。时间周期已用完，将3号程

**第10行：“1”改为 “ 2 ”：**

（48）获取当前指令，第2个程序的第p[2]（p[2]=6）单元，当前运行指令为cur=“print b”，该指令是第2个程序中的输出，val[1]=8，因此输出2:8。p[2]++，此时p[2]=7，指向第2个程

1. **页码:183　　行数:3　　印次:1**

**“1”改为 “ 3 ”：**

（50）获取当前指令，第3个程序的第p[3]（p[3]=5）单元，当前运行指令为cur=“b = 21”，令val[1]=21，p[3]++，此时p[3]=6，指向第3个程序的下一条指令。时间周期已用完，将3号

1. **页码:324　　行数:2,11,13　　印次:1**

**第2行：“dist[0][2]”改为 “ dist[2][3] ”：**

* dist[2][3]：dist[2][3]=8>dist[2][1]+dist[1][3]=6，更新dist[2][3]=6，*p*[2][3]=1。在节点2、3之间插入节点1。

**第11,13行：“dist[2][1]”改为 “ dist[2][0] ”，p改为斜体：**

* dist[3][0]：dist[3][0]=∞>dist[3][2]+dist[2][0]=9，更新dist[3][0]=9，*p*[3][0]=2。在节点3、0之间插入节点2。
* dist[3][1]：dist[3][1]=∞>dist[3][2]+dist[2][1]=10，更新dist[3][1]=10，*p*[3][1]=*p*[2][1]=0。在节点3、1之间插入节点2。

1. **页码:337　　行数:-3　　印次:1**

**“*V*”改为 “*U* ”：**

现在只需看看在连接两个集合（*U*和*V*−*U*）的边中，哪一条边的权值最小，把权值最小的

1. **页码:338　　行数:1　　印次:1**

**“*V*”改为 “*U* ”：**

再从连接两个集合（*U*和*V*−*U*）的边中选择一条权值最小的边。从上图可以看出，在连接

1. **页码:382　　行数:24　　印次:1**

**“HDU4019”改为 “HDU4109 ”：**

1. **页码:458　　行数:-7　　印次:1**

**“TOJ3374”改为 “SDUTOJ3374”：**

1. **页码:474　　行数:20　　印次:1**

**judge函数中添加“**||dis[n+1]-dis[cur]<x**”：**

if(cur>n||dis[n+1]-dis[cur]<x)

return 0; //如果在这个过程中大于n了，说明放不开