```
Process arrival runtime
1       0       4
2       7       10
3       9       2
4       13      8
5       25      1
6       35      7
7       36      2
First come first served...
Process 1 started at time 0 and finished at time 4
Process 2 started at time 7 and finished at time 17
Process 3 started at time 17 and finished at time 19
Process 4 started at time 19 and finished at time 27
Process 5 started at time 27 and finished at time 28
Process 6 started at time 35 and finished at time 42
Process 7 started at time 42 and finished at time 44
Shortest job first...
Process 1 started at time 0 and finished at time 4
Process 2 started at time 7 and finished at time 17
Process 3 started at time 17 and finished at time 19
Process 4 started at time 19 and finished at time 27
Process 5 started at time 27 and finished at time 28
Process 6 started at time 35 and finished at time 42
Process 7 started at time 42 and finished at time 44
```

```
17 void first_come_first_served()
18 {
19   /* Your code for FCFS algorithm here */
20
21   int starttime = 0;
22   int endtime = 0;
23   for (int i = 0; i < 101; i++)
24   {
25     if (proc[i].runtime != 0)
26     { //excludes any iteration of the struct outside of where we have data.
27       starttime = endtime;
28
29       if (proc[i].arrivaltime == 0)
30       { //first process arriving defines initial endtime
31         endtime = starttime + proc[i].runtime;
32       }
33
34       if (proc[i].arrivaltime >= endtime)
35       { //(processor idles) update starttime to be proc[i].arrivaltime
36         starttime = proc[i].arrivaltime;
37       }
38
39       endtime = starttime + proc[i].runtime;
40       printf("Process %d started at time %d and finished at time %d\n", proc[i].id, starttime, endtime);
41     }
42   }
43 }
```

```
45 void shortest_job_first()
46 {
47   /* Your code for SJF algorithm here */
48   int endtime = proc[0].arrivaltime;
49   int starttime;
50   int lowestRunTime = proc[0].runtime;
51   int nextToRun = proc[0].id;
52   int runableProc[100];
53
54   for (int z = 0; z < 101; z++)
55   {
56     runableProc[z] = 102;
57   }
58
59   for (int x = 0; x < 101; x++)
60   {
61
62     int arrayCheck = 0;
63
64     if (proc[x].runtime != 0)
65     { //excludes any iteration of the struct outside of where we have data.
66
67       for (int i = 0; i < 101; i++)
68       { //checks what is runnable and what is not for a given iteration
69         if (proc[i].runtime != 0)
70         { //excludes any iteration of the struct outside of where we have data.
71           if (proc[i].arrivaltime <= endtime)
72           { //what procs are available
73             if (!(proc[i].endtime > 0))
74             { // disregards procs that have run
75
76               runableProc[i] = proc[i].id; //adds current proc to temp array
77             }
78           }
79           else
80           {
81             runableProc[i] = 102; //assign index as 102 so we can check for it later
82           }
83         }
84       }
85
86       for (int y = 0; y < 101; y++)
87       {
88         if (runableProc[y] != 102)
89         {
90           arrayCheck = 1;
91         }
92       }
93
94       for (int j = 0; j < 101; j++)
95       { //which process to run next
96         if (proc[j].runtime != 0)
97         { //excludes any iteration of the struct outside of where we have data.
```

```
 99          if (runableProc[j] != 102)
100          { //checking to see what processes can be chosen from
101
102            if (proc[j].runtime < lowestRunTime)
103            {
104              lowestRunTime = proc[j].runtime;
105              nextToRun = proc[j].id;
106            }
107            else
108            {
109              if (proc[j].arrivaltime == proc[nextToRun - 1].arrivaltime)
110              {
111                if (proc[j].runtime < proc[nextToRun - 1].runtime)
112                {
113                  lowestRunTime = proc[j].runtime;
114                  nextToRun = proc[j].id;
115                }
116              }
117            }
118          }
119        }
120      }
121      if (arrayCheck == 0)
122      {
123        lowestRunTime = proc[x].runtime;
124        nextToRun = proc[x].id;
125        endtime = proc[x].arrivaltime;
126      }
127
128      runableProc[nextToRun - 1] = 102;
129
130      starttime = endtime;
131      endtime = endtime + lowestRunTime;
132      lowestRunTime = endtime;
133      proc[nextToRun - 1].endtime = endtime;
134      printf("Process %d started at time %d and finished at time %d\n", nextToRun, starttime, endtime);
135    }
136  }
137 }
```