

```

1. #include<stdio.h>
2. #include<string.h>
3.
4. struct proc_struct {
5.     char proc_name[20];
6.     int arr_time, bur_time, comp_time, rem;
7. }temp_Struct;
8.
9.
10. void faculty_Queue(int no_of_process) {
11.
12.     int count, arr_time, bur_time, quan_time;
13.     struct proc_struct faculty_proc[no_of_process];
14.
15.     for(count = 0; count < no_of_process; count++) {
16.         printf("Enter the details of Process[%d]", count+1);
17.         puts("");
18.         printf("Process Name : ");
19.         scanf("%s", faculty_proc[count].proc_name);
20.
21.         printf("Arrival Time : ");
22.         scanf("%d", &faculty_proc[count].arr_time);
23.
24.         printf("Burst Time : ");
25.         scanf("%d", &faculty_proc[count].bur_time);
26.         puts("");
27.     }
28.     printf("Now, enter the quantum time for FACULTY queue : ");
29.     scanf("%d", &quan_time);
30.
31.
32.     // sorting the processes by their ARRIVAL time.
33.     // if the ARRIVAL time is same then scheduling is based on FCFS.
34.     for(count = 0; count < no_of_process; count++) {
35.         for(int x = count +1; x < count; x++){
36.             if(faculty_proc[count].arr_time > faculty_proc[x].arr_time) {
37.                 temp_Struct = faculty_proc[count];
38.                 faculty_proc[count] = faculty_proc[x];
39.                 faculty_proc[x] = temp_Struct;
40.             }
41.         }
42.     }
43.
44.     // initially all the burst time is rem and completion of process is zero.
45.     for(count = 0; count < no_of_process; count++) {
46.         faculty_proc[count].rem = faculty_proc[count].bur_time;
47.         faculty_proc[count].comp_time = 0;
48.     }
49.
50.     int total_time, queue, round_robin[20];
51.     total_time = 0;
52.     queue = 0;
53.     round_robin[queue] = 0;
54.
55.
56.     int flag, x, n, z, waiting_time = 0;
57.     do {
58.         for(count = 0; count < no_of_process; count++){
59.             if(total_time >= faculty_proc[count].arr_time){
60.                 z = 0;
61.                 for(x = 0; x <= queue; x++) {
62.                     if(round_robin[x] == count) {
63.                         z++;
64.                     }
65.                 }
66.                 if(z == 0) {

```

```

67.         queue++;
68.         round_robin[queue] == count;
69.     }
70. }
71. }
72.
73. if(queue == 0) {
74.     n = 0;
75. }
76. if(faculty_proc[n].rem == 0) {
77.     n++;
78. }
79. if(n > queue) {
80.     n = (n - 1) % queue;
81. }
82. if(n <= queue) {
83.     if(faculty_proc[n].rem > 0) {
84.         if(faculty_proc[n].rem < quan_time){
85.             total_time += faculty_proc[n].rem;
86.             faculty_proc[n].rem = 0;
87.         }else {
88.             total_time += quan_time;
89.             faculty_proc[n].rem -= quan_time;
90.         }
91.         faculty_proc[n].comp_time = total_time;
92.     }
93.     n++;
94. }
95. flag = 0;
96.
97. for(count = 0; count < no_of_process; count++) {
98.     if(faculty_proc[count].rem > 0) {
99.         flag++;
100.    }
101. }
102. }while(flag != 0);
103.
104.
105. puts("\n\t\t\t*****");
106. puts("\t\t\t*****  ROUND ROBIN ALGORITHM OUTPUT  *****");
107. puts("\t\t\t*****\n");
108. printf("\n\t\tProcess Name\t\t\t\t\tArrival Time\t\t\t\t\tBurst Time\t\t\t\t\tCompletion
Time\t\t\t\t\t");
109.
110. for(count = 0; count < no_of_process; count++){
111.     waiting_time = faculty_proc[count].comp_time - faculty_proc[count].bur_time
- faculty_proc[count].arr_time;
112.
113.     printf("\n\t\t %s\t\t\t\t\t %d\t\t\t\t\t %d\t\t\t\t\t %d\t\t\t\t\t \n", faculty_pr
oc[count].proc_name, faculty_proc[count].arr_time, faculty_proc[count].bur_time, faculty_pro
c[count].comp_time);
114. }
115.
116. }
117.
118.
119. void student_Queue(int no_of_process) {
120.
121.     int count, arr_time, bur_time, quan_time;
122.     struct proc_struct student_proc[no_of_process];
123.
124.     for(count = 0; count < no_of_process; count++) {
125.         printf("Enter the details of Process[%d]", count+1);
126.         puts("");
127.         printf("Process Name : ");
128.         scanf("%s", student_proc[count].proc_name);

```

```

129.
130.         printf("Arrival Time : ");
131.         scanf("%d", &student_proc[count].arr_time);
132.
133.         printf("Burst Time : ");
134.         scanf("%d", &student_proc[count].bur_time);
135.     }
136.     printf("Now, enter the quantum time for STUDENT queue : ");
137.     scanf("%d", &quan_time);
138.
139.
140.     // sorting the processes by their ARRIVAL time.
141.     // if the ARRIVAL time is same then scheduling is based on FCFS.
142.     for(count = 0; count < no_of_process; count++) {
143.         for(int x = count +1; x < count; x++){
144.             if(student_proc[count].arr_time > student_proc[x].arr_time) {
145.                 temp_Struct = student_proc[count];
146.                 student_proc[count] = student_proc[x];
147.                 student_proc[x] = temp_Struct;
148.             }
149.         }
150.     }
151.
152.     // initially all the burst time is rem and completion of process is zero.
153.     for(count = 0; count < no_of_process; count++) {
154.         student_proc[count].rem = student_proc[count].bur_time;
155.         student_proc[count].comp_time = 0;
156.     }
157.
158.     int total_time, queue, round_robin[20];
159.     total_time = 0;
160.     queue = 0;
161.     round_robin[queue] = 0;
162. }
163.
164.
165. int main(int argc, char const *argv[]) {
166.     int select_queue, no_of_process;
167.
168.     puts("Please choose a queue to post your query : ");
169.     puts("1. FACULTY queue.");
170.     puts("2. STUDENT queue.");
171.     printf("> ");
172.     scanf("%d", &select_queue);
173.
174.     switch(select_queue) {
175.         case 1 :
176.             printf("Enter number of process for FACULTY queue : ");
177.             scanf("%d", &no_of_process);
178.
179.             faculty_Queue(no_of_process);
180.
181.             break;
182.
183.         case 2 :
184.             printf("Enter number of process for STUDENT queue : ");
185.             scanf("%d", &no_of_process);
186.
187.             student_Queue(no_of_process);
188.
189.             break;
190.
191.         default :
192.             printf("Please select the correct option by running the program
again.");
193.     }

```

```
194.  
195.     return 0;  
196. }
```