```cpp
#include<bits/stdc++.h>
using namespace std;
struct process {

    pid_t p_no = 0;
    time_t start_AT = 0, AT = 0,
        BT_left = 0, BT = 0, temp_BT = 0,
        CT = 0, TAT = 0, WT = 0, RT = 0;
    int priority = 0;

    void set_CT(time_t time)
    {
        CT = time;
        set_TAT();
        set_WT();
    }

    void set_TAT()
    {
        TAT = CT - start_AT;
    }

    void set_WT()
    {
        WT = TAT - BT;
    }
    void P_set()
    {
        start_AT = AT;
        BT_left = BT;
    }
    void set_RT(time_t time)
    {
        RT = time - start_AT;
    }


    friend bool operator<(const process& a, const process& b)
    {
        return a.AT > b.AT;
    }
};

process pop_index(priority_queue<process>* main_queue,
,               int index)
{
    priority_queue<process> rm_index;
    int i;
    process p;

    switch (index) {
    case 0:
        p = (*main_queue).top();
        (*main_queue).pop();
        break;
    default:
        for (i = 0; i < index; i++) {
            rm_index.push((*main_queue).top());
            (*main_queue).pop();
        }
        p = (*main_queue).top();
```

```
62          (*main_queue).pop();
63
64          while (!(*main_queue).empty()) {
65              rm_index.push((*main_queue).top());
66              (*main_queue).pop();
67          }
68          (*main_queue) = rm_index;
69          break;
70      }
71      return p;
72  }
73
74  int max_response_ratio_index(priority_queue<process> ready_queue,
75  ,                            time_t limit)
76  {
77      int index, i = 0;
78      double response_ratio = 0, max = 0;
79
80      while (!ready_queue.empty()
81          && ready_queue.top().AT <= limit) {
82          response_ratio = ((double)(limit - ready_queue.top().AT) +
    ready_queue.top().BT_left) / ready_queue.top().BT_left;
83          if (response_ratio > max) {
84              max = response_ratio;
85              index = i;
86          }
87          i++;
88          ready_queue.pop();
89      }
90
91      return index;
92  }
93
94  priority_queue<process> HRRN_run(priority_queue<process> ready_queue,
95  ,                                queue<process>* gantt)
96  {
97      priority_queue<process> completion_queue;
98      process p;
99      time_t clock = 0;
100
101     while (!ready_queue.empty()) {
102         while (clock < ready_queue.top().AT) {
103             p.temp_BT++;
104             clock++;
105         }
106         if (p.temp_BT > 0) {
107             p.p_no = -1;
108             p.CT = clock;
109             (*gantt).push(p);
110         }
111         p = pop_index(&ready_queue,
112                       max_response_ratio_index(ready_queue,
113                                                clock));
114         p.set_RT(clock);
115
116         while (p.BT_left > 0) {
117             p.temp_BT++;
118             p.BT_left--;
119             clock++;
120         }
121         p.set_CT(clock);
```

```
122         (*gantt).push(p);
123         p.temp_BT = 0;
124 
125         completion_queue.push(p);
126     }
127     return completion_queue;
128 }
129 
130 priority_queue<process> set_sample_data()
131 {
132     priority_queue<process> ready_queue;
133     int n;
134     cout<<"\nEnter the number of processes: "; cin>>n;
135 
136     for(int i=0;i<n;i++)
137     {
138           cout<<"\nEnter arrival time ,burst time and priority of process "<<i+1<<"
    : ";
139           process temp;
140             cin>>temp.AT>>temp.BT>>temp.priority;
141             temp.p_no = i+1;
142             temp.P_set();
143             ready_queue.push(temp);
144 
145     }
146     cout<<"\n";
147 
148     return ready_queue;
149 }
150 
151 double get_total_WT(priority_queue<process> processes)
152 {
153     double total = 0;
154     while (!processes.empty()) {
155         total += processes.top().WT;
156         processes.pop();
157     }
158     return total;
159 }
160 
161 double get_total_TAT(priority_queue<process> processes)
162 {
163     double total = 0;
164     while (!processes.empty()) {
165         total += processes.top().TAT;
166         processes.pop();
167     }
168     return total;
169 }
170 
171 double get_total_CT(priority_queue<process> processes)
172 {
173     double total = 0;
174     while (!processes.empty()) {
175         total += processes.top().CT;
176         processes.pop();
177     }
178     return total;
179 }
180 
181 double get_total_RT(priority_queue<process> processes)
```

```
182 {
183     double total = 0;
184     while (!processes.empty()) {
185         total += processes.top().RT;
186         processes.pop();
187     }
188     return total;
189 }
190 
191 void disp(priority_queue<process> main_queue, bool high)
192 {
193     int i = 0, temp, size = main_queue.size();
194     priority_queue<process> tempq = main_queue;
195     double temp1;
196     cout << "+-------------+--------------";
197     cout << "+------------+-----------------";
198     cout << "+-----------------+--------------+---------------+";
199     if (high == true)
200         cout << "----------+" << endl;
201     else
202         cout << endl;
203     cout << "| Process No. | Arrival Time ";
204     cout << "| Burst Time | Completion Time ";
205     cout << "| Turnaround Time | Waiting Time | Response Time |";
206     if (high == true)
207         cout << " Priority |" << endl;
208     else
209         cout << endl;
210     cout << "+-------------+--------------";
211     cout << "+------------+-----------------";
212     cout << "+-----------------+--------------+---------------+";
213     if (high == true)
214         cout << "----------+" << endl;
215     else
216         cout << endl;
217     while (!main_queue.empty()) {
218         temp = to_string(main_queue.top().p_no).length();
219         cout << '|' << string(6 - temp / 2 - temp % 2, ' ')
220             << main_queue.top().p_no << string(7 - temp / 2, ' ');
221         temp = to_string(main_queue.top().start_AT).length();
222         cout << '|' << string(7 - temp / 2 - temp % 2, ' ')
223             << main_queue.top().start_AT << string(7 - temp / 2, ' ');
224         temp = to_string(main_queue.top().BT).length();
225         cout << '|' << string(6 - temp / 2 - temp % 2, ' ')
226             << main_queue.top().BT << string(6 - temp / 2, ' ');
227         temp = to_string(main_queue.top().CT).length();
228         cout << '|' << string(8 - temp / 2 - temp % 2, ' ')
229             << main_queue.top().CT << string(9 - temp / 2, ' ');
230         temp = to_string(main_queue.top().TAT).length();
231         cout << '|' << string(8 - temp / 2 - temp % 2, ' ')
232             << main_queue.top().TAT << string(9 - temp / 2, ' ');
233         temp = to_string(main_queue.top().WT).length();
234         cout << '|' << string(7 - temp / 2 - temp % 2, ' ')
235             << main_queue.top().WT << string(7 - temp / 2, ' ');
236         temp = to_string(main_queue.top().RT).length();
237         cout << '|' << string(7 - temp / 2 - temp % 2, ' ')
238             << main_queue.top().RT << string(8 - temp / 2, ' ');
239         if (high == true) {
240             temp = to_string(main_queue.top().priority).length();
241             cout << '|' << string(5 - temp / 2 - temp % 2, ' ')
242                 << main_queue.top().priority << string(5 - temp / 2, ' ');
```

```
243         }
244         cout << "|\n";
245         main_queue.pop();
246     }
247     cout << "+-------------+--------------";
248     cout << "+------------+-----------------";
249     cout << "+-----------------+--------------+--------------+";
250     if (high == true)
251         cout << "----------+";
252     cout << endl;
253     temp1 = get_total_CT(tempq);
254     cout << "\nTotal completion time :- " << temp1 << endl;
255     cout << "Average completion time :- " << temp1 / size << endl;
256     temp1 = get_total_TAT(tempq);
257     cout << "\nTotal turnaround time :- " << temp1 << endl;
258     cout << "Average turnaround time :- " << temp1 / size << endl;
259     temp1 = get_total_WT(tempq);
260     cout << "\nTotal waiting time :- " << temp1 << endl;
261     cout << "Average waiting time :- " << temp1 / size << endl;
262     temp1 = get_total_RT(tempq);
263     cout << "\nTotal response time :- " << temp1 << endl;
264     cout << "Average response time :- " << temp1 / size << endl;
265 }
266
267 void disp_gantt_chart(queue<process> gantt)
268 {
269     int temp, prev = 0;
270     queue<process> spaces = gantt;
271     cout << "\n\nGantt Chart (IS indicates ideal state) :- \n\n+";
272     while (!spaces.empty()) {
273         cout << string(to_string(spaces.front().p_no).length() +
(spaces.front().p_no != -1) + 2 * spaces.front().temp_BT, '-') << "+";
274         spaces.pop();
275     }
276     cout << "\n|";
277     spaces = gantt;
278     while (!spaces.empty()) {
279         cout << string(spaces.front().temp_BT, ' ');
280         if (spaces.front().p_no == -1)
281             cout << "IS" << string(spaces.front().temp_BT, ' ') << '|';
282         else
283             cout << "P" << spaces.front().p_no
284                 << string(spaces.front().temp_BT, ' ') << '|';
285         spaces.pop();
286     }
287     spaces = gantt;
288     cout << "\n+";
289     while (!spaces.empty()) {
290         cout << string(to_string(spaces.front().p_no).length() +
(spaces.front().p_no != -1) + 2 * spaces.front().temp_BT, '-')
291             << "+";
292         spaces.pop();
293     }
294     spaces = gantt;
295     cout << "\n0";
296     while (!spaces.empty()) {
297         temp = to_string(spaces.front().CT).length();
298         cout << string(to_string(spaces.front().p_no).length() +
(spaces.front().p_no != -1) + 2 * spaces.front().temp_BT - temp / 2 - prev, ' ')
299             << spaces.front().CT;
300         prev = temp / 2 - temp % 2 == 0;
301         spaces.pop();
```

```
302     }
303     cout << "\n\n";
304 }
305 
306 int main()
307 {
308     priority_queue<process> ready_queue, completion_queue;
309 
310     queue<process> gantt;
311     ready_queue = set_sample_data();
312 
313     completion_queue = HRRN_run(ready_queue, &gantt);
314 
315     disp(completion_queue, false);
316 
317     disp_gantt_chart(gantt);
318     return 0;
319 }
```

```
Enter the number of processes: 7

Enter arrival time ,burst time and priority of process 1 : 0 4 2

Enter arrival time ,burst time and priority of process 2 : 1 2 4

Enter arrival time ,burst time and priority of process 3 : 2 3 6

Enter arrival time ,burst time and priority of process 4 : 3 5 10

Enter arrival time ,burst time and priority of process 5 : 4 1 8

Enter arrival time ,burst time and priority of process 6 : 5 4 12

Enter arrival time ,burst time and priority of process 7 : 6 6 9

+------------+--------------+------------+-----------------+-----------------+--------------+---------------+
| Process No. | Arrival Time | Burst Time | Completion Time | Turnaround Time | Waiting Time | Response Time |
+------------+--------------+------------+-----------------+-----------------+--------------+---------------+
|     1      |      0       |     4      |        4        |        4        |      0       |       0       |
|     2      |      1       |     2      |        6        |        5        |      3       |       3       |
|     3      |      2       |     3      |       10        |        8        |      5       |       5       |
|     4      |      3       |     5      |       15        |       12        |      7       |       7       |
|     5      |      4       |     1      |        7        |        3        |      2       |       2       |
|     6      |      5       |     4      |       19        |       14        |      10      |       10      |
|     7      |      6       |     6      |       25        |       19        |      13      |       13      |
+------------+--------------+------------+-----------------+-----------------+--------------+---------------+

Total completion time :- 86
Average completion time :- 12.2857

Total turnaround time :- 65
Average turnaround time :- 9.28571

Total waiting time :- 40
Average waiting time :- 5.71429

Total response time :- 40
Average response time :- 5.71429


Gantt Chart (IS indicates ideal state) :-

+----------+------+----+--------+------------+----------+--------------+
|    P1    |  P2  | P5 |   P3   |     P4     |    P6    |      P7      |
+----------+------+----+--------+------------+----------+--------------+
0          4      6    7       10           15         19             25


anzal  anzal  ../4th sem/os lab/p8
```