```cpp
#include<bits/stdc++.h>
using namespace std;
int bufftime=9999;
vector<int>freq;
int flag =0;
int res=0;

    // int process_id[] = {0, 1, 2, 3, 4, 5, 6};
    // int arrival_time[] = {9, 1, 4, 5, 2, 30, 29};
    // int burst_time[] = {10, 2, 1, 5, 7, 3, 6};

class Process{
    public:
        int process_id;
        int arrival_time;
        int burst_time;
        int remaining_time;
        int completion_time = 0;
        int turn_around_time = -1;
        int waiting_time = -1;
        int response_time=-1;
        int start_time=0;
        int prev=0;

        Process(int p_id, int a_time, int b_time){
            this->process_id = p_id;
            this->arrival_time = a_time;
            this->burst_time = b_time;
            this->remaining_time = this->burst_time;
        }

        bool operator!=(Process& p){
            if (this->process_id == p.process_id) return false;
            return true;
        }
};

bool idSort(Process *a,Process *b){
    return a->process_id < b->process_id;
}

bool idexist(vector<Process>response,int id)
{   for(Process p:response)
{
  if(id==p.process_id)
  {
      return false;
  }
}
return true;
}

void print_table(int m,vector<Process> p,vector<Process>presp)
{
    int i;

    puts("+-----+------------+--------------+-----------------+-----------------+--
--------------+-----------------+");
    puts("| PID | Burst Time | Arrival Time | Turnaround Time |   Waiting Time  |
Completion Time |  Response Time  |");
    puts("+-----+------------+--------------+-----------------+-----------------+--
```

```
---------------+-----------------+");

    for(i=0; i<m; i++) {
        printf("| %2d  |     %2d     |      %2d      |        %2d       |
%2d       |        %2d       |        %2d       |\n"
               , p[i].process_id, p[i].burst_time, p[i].arrival_time,
p[i].turn_around_time,  p[i].waiting_time, p[i].completion_time,
presp[i+res].response_time );
        puts("+-----+------------+--------------+-----------------+---------------
-+-----------------+-----------------+");
    }
    cout<<"\n";
}

void print_gantt_chart(vector<Process> gantt)
{   int y=gantt.size();
    cout<<"\n";
    int i, j;
    if(bufftime==9999)
    {
        bufftime=0;
    }
    if(bufftime!=0){
    //-----------------------------------------------
    printf(" ");
    // for(i=0; i<n; i++) {
        for(j=0; j<bufftime; j++) printf("--");
        printf(" ");
    // }
    // printf("\n|");
    //-----------------------------------------------
    }

    if(bufftime==0){
    printf(" ");
    }
    for(i=flag; i<y; i++) {
        for(j=0; j<freq[i]; j++) printf("--");
        printf(" ");
    }
    printf("\n|");

    if(bufftime!=0){
    //-----------------------------------------------
    // for(i=0; i<n; i++) {
        for(j=0; j<bufftime - 1; j++) printf(" ");
        printf("I%d", bufftime);
        for(j=0; j<bufftime - 1; j++) printf(" ");
        printf("|");
    // }
    // printf("\n ");
    //-----------------------------------------------
    }


    for(i=flag; i<y; i++) {
        for(j=0; j<freq[i] - 1; j++) printf(" ");
        if(gantt[i].process_id!=-1) {printf("P%d", gantt[i].process_id);}
        else {printf("IL");}
        for(j=0; j<freq[i] - 1; j++) printf(" ");
        printf("|");
    }
```

```
117     printf("\n ");
118 
119     if(bufftime!=0){
120     //-----------------------------------------------
121     // for(i=0; i<n; i++) {
122         for(j=0; j<bufftime; j++) printf("--");
123         printf(" ");
124     // }
125     // printf("\n");
126     //-----------------------------------------------
127     }
128 
129 
130     for(i=flag; i<y; i++) {
131         for(j=0; j<freq[i]; j++) printf("--");
132         printf(" ");
133     }
134     printf("\n");
135 
136     if(bufftime!=0){
137     //-----------------------------------------------
138     printf("0");
139     // for(i=0; i<n; i++) {
140         for(j=0; j<bufftime; j++) printf("  ");
141         // if(bufftime > 9) printf("\b");
142         // printf("%d", bufftime);
143 
144     // }
145     // printf("\n");
146     //-----------------------------------------------
147     }
148     if(gantt[i].completion_time!=0)
149     {
150         printf("0");
151     }
152     printf("%d",bufftime);
153     for(i=flag; i<y; i++) {
154         for(j=0; j<freq[i]; j++) printf("  ");
155         if(gantt[i].completion_time > 9) printf("\b");
156         printf("%d", gantt[i].completion_time);
157 
158     }
159     printf("\n");
160 
161 }
162 
163 int main(){
164     int n;
165     int tq;
166     cout<<"\nEnter the no of processes :";
167     cin>>n;
168     cout<<"\nEnter the time quantum :";
169     cin>>tq;
170     int process_id[n] ;
171     int arrival_time[n] ;
172     int burst_time[n] ;
173     cout<<"\nEnter the  process id :";
174     for(int i=0;i<n;i++)
175     {
176       cin>>process_id[i];
177     }
```

```
178     cout<<"\nEnter the arrival time for each process :";
179         for(int i=0;i<n;i++)
180     {
181         cin>>arrival_time[i];
182     }
183     cout<<"\nEnter the burst time for each process :";
184         for(int i=0;i<n;i++)
185     {
186         cin>>burst_time[i];
187     }
188 
189 
190     vector<Process> process_list;
191 
192     queue<Process*> ready_queue;
193     vector<Process> gantt;
194     vector<Process> response;
195     vector<Process> prinresp;
196 
197     Process idle(-1,0,0);
198 
199 
200     for(int i=0; i<n; i++)
201         process_list.push_back(Process(process_id[i], arrival_time[i],
    burst_time[i]));
202 
203 
204 // Bubble sort on arrival_time:
205     for(int i=0; i<n; i++)
206         for (int j=1; j<n-i; j++)
207             if (process_list.at(j).arrival_time < process_list.at(j-
    1).arrival_time){
208                 Process temp = process_list.at(j);
209                 process_list.at(j) = process_list.at(j-1);
210                 process_list.at(j-1)= temp;
211             }
212 
213     if(process_list.at(0).arrival_time > 0)
214         {idle.completion_time = process_list.at(0).arrival_time;
215 
216         }
217         else{flag=1;}
218     gantt.push_back(idle);
219 
220     ready_queue.push(&process_list.at(0));
221 
222     while( !ready_queue.empty() ){
223         Process* current = ready_queue.front();
224         ready_queue.pop();
225 
226         if(current->remaining_time <= tq){
227             current->completion_time = gantt.back().completion_time + current-
    >remaining_time;
228             current->remaining_time = 0;
229         }
230         else{
231             current->completion_time = gantt.back().completion_time + tq;
232             current->remaining_time = current->remaining_time - tq;
233         }
234 
235 
236 
```

```
237         for(Process& p: process_list){
238             if(p.arrival_time > gantt.back().completion_time && p.arrival_time <=
    current->completion_time ){
239                 ready_queue.push(&p);
240 
241             }
242         }
243 
244         if(current->remaining_time != 0)
245             ready_queue.push(current);
246 
247         gantt.push_back(*current);
248 
249 
250         if (ready_queue.empty() )
251             for (Process& p: process_list)
252                 if(p.remaining_time != 0 ){
253                     idle.arrival_time = gantt.back().completion_time;
254                     idle.completion_time = p.arrival_time;
255                     gantt.push_back(idle);
256                     ready_queue.push(&p);
257 
258                     break;
259                 }
260 
261     }
262 
263 
264     // cout << "CPU Scheduling:" << endl << "Round Robin Scheduling:";
265 
266     // printf("\n\n %15s | %15s | %15s | %15s | %15s | %15s |\n\n", "Process Id",
267     //                 "Arrival Time",
268     //                 "Burst Time",
269     //                 "Completion Time",
270     //                 "Turn Around T.",
271     //                 "Waiting Time");
272 
273 // Bubble sort on process_id:
274 
275 //  for(int i=0; i<n; i++)
276 //      for (int j=1; j<n-i; j++)
277 //          if (process_list.at(j).process_id < process_list.at(j-1).process_id){
278 //              Process temp = process_list.at(j);
279 //              process_list.at(j) = process_list.at(j-1);
280 //              process_list.at(j-1) = temp;
281 //          }
282     float total_turnaround_time=0.0;
283     float total_waiting_time=0.0;
284     float total_response_time=0.0;
285     float total_completion_time=0.0;
286 
287 
288      for(Process p:gantt)
289    {
290 
291          response.push_back(p);
292 
293 
294    }
295 
296    for(int i=0;i<response.size();i++)
```

```
297    {
298         if(response[i].process_id==-1)
299         {
300             res=1;
301         }
302         else{
303             response[i].response_time=response[i-1].completion_time-
response[i].arrival_time;
304         }
305    }
306 
307      for(Process p:response)
308     {
309    if(idexist(prinresp,p.process_id))
310    {
311         prinresp.push_back(p);
312         if(p.response_time!=-1)
313         {
314 
315         total_response_time += p.response_time;
316         }
317    }
318    }
319 
320     reverse( gantt.begin(), gantt.end() );
321 
322     for(Process& p: process_list){
323         p.turn_around_time = p.completion_time - p.arrival_time;
324         p.waiting_time = p.turn_around_time - p.burst_time;
325         p.response_time=p.start_time-p.arrival_time;
326         total_completion_time += p.completion_time;
327         total_turnaround_time += p.turn_around_time;
328         total_waiting_time += p.waiting_time;
329 
330         // printf("  %14d |  %14d |  %14d |  %14d |  %14d |  %14d |\n",
p.process_id,
331         //                   p.arrival_time,
332         //                   p.burst_time,
333         //                   p.completion_time,
334         //                   p.turn_around_time,
335         //                   p.waiting_time);
336     }
337     cout<<"\n";
338     cout<<"\n";
339     cout<<"Table"<<"\n";
340     print_table(process_list.size(),process_list,prinresp);
341     cout<<"\n";
342 
343 
344     reverse( gantt.begin(), gantt.end() );
345 
346 
347 
348 
349 
350     // for (Process& p: gantt)
351     // {
352     //  cout<<"---- ";
353     // }
354     // cout<<endl;
355     // for (Process& p: gantt)
356     //  if (p.process_id == -1){
```

```
357     //      if (p.completion_time != 0)
358     //          printf("   idle |");
359     //  }
360     //  else
361     //      printf("     P%d |", p.process_id);
362 
363 
364     // cout << endl << 0;
365 
366     // for (Process& p: gantt)
367     //  if (p.completion_time)
368     //      printf("     %2d |", p.completion_time);
369 
370     // cout<<endl;
371 
372     // for (Process& p: gantt)
373     // {
374     //  cout<<"---- ";
375     // }
376 
377 
378 
379 
380 
381 
382 
383 
384     for(int i=0;i<gantt.size();i++)
385     {
386 
387 
388       if(i==0)
389       {
390          freq.push_back(gantt[i].completion_time);
391       }
392       else{
393           freq.push_back(gantt[i].completion_time-gantt[i-1].completion_time);
394       }
395 
396     }
397 
398 //   for(int x: freq)
399 //   {
400 //    cout<<x<<" ";
401 //   }
402   int a =process_list.size();
403 
404     // cout<<"\n";
405     cout<<"Average TurnAround time : "<<(total_turnaround_time/a)<<"\n";
406     cout<<"Average Waiting time : "<<(total_waiting_time/a)<<"\n";
407     cout<<"Average Response time : "<<(total_response_time/a)<<"\n";
408     cout<<"Average Completion time : "<<(total_completion_time/a)<<"\n";
409     cout << "\n\nGantt Chart:\n ";
410     cout<<"\n";
411     print_gantt_chart(gantt);
412 
413 
414 
415 
416     // sort(response.begin(),response.end(),idSort);
417 
```

```
418
419
420
421
422     return 0;
423
424 }
```

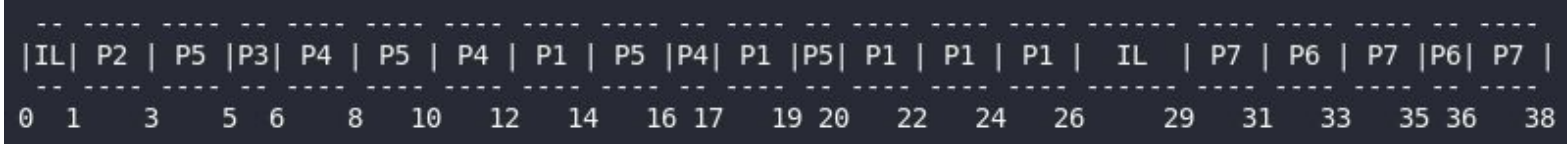
```
Enter the no of processes :7

Enter the time quantum :2

Enter the  process id :1 2 3 4 5 6 7

Enter the arrival time for each process :9 1 4 5 2 30 29

Enter the burst time for each process :10 2 1 5 7 3 6
```

Table

PID	Burst Time	Arrival Time	Turnaround Time	Waiting Time	Completion Time	Response Time
2	2	1	2	0	3	0
5	7	2	18	11	20	1
3	1	4	2	1	6	1
4	5	5	12	7	17	1
1	10	9	17	7	26	3
7	6	29	9	3	38	0
6	3	30	6	3	36	1

```
Average TurnAround time : 9.42857
Average Waiting time : 4.57143
Average Response time : 1
Average Completion time : 20.8571
```

Gantt Chart:




```
anzal  anzal  ../4th sem/os lab/p5
```