

**Project Topic :-**  
**Faculty Query- Time Scheduler**

[USing RR Algorithm]

Submitted for:-  
**Operating System (UCS303)**

Submitted To:-  
**Dr. Shashank S. Singh Sir**

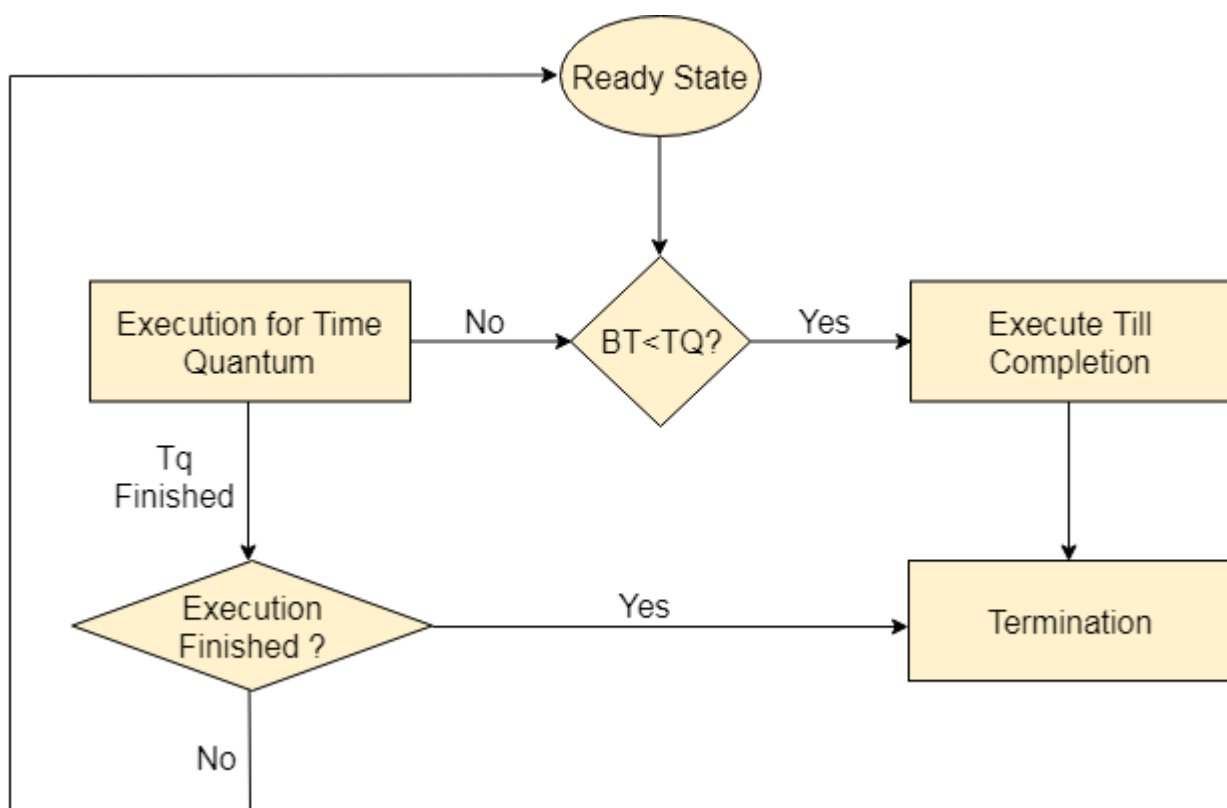
Submitted By:-  
**Aryan Varshney – 101903605**  
**Rohan Kumar -101903625**



**THAPAR INSTITUTE OF ENGINEERING & TECHNOLOGY,**  
**(Deemed To Be University), Patiala, Punjab, India**

## Introduction :-

Round Robin scheduling algorithm is one of the most popular scheduling algorithm which can actually be implemented in most of the operating systems. This is the **pre-emptive version** of first come first serve scheduling. The Algorithm focuses on Time Sharing. In this algorithm, every process gets executed in a **cyclic way**. A certain time slice is defined in the system which is called time **quantum**. Each process present in the ready queue is assigned the CPU for that time quantum, if the execution of the process is completed during that time then the process will **terminate** else the process will go back to the **ready queue** and waits for the next turn to complete the execution.



## **Problem Statement :-**

" Dr. Shashank Sir is a Linux expert who wants to have an online system where he can handle student queries. Since there can be multiple requests at any time he wishes to dedicate a fixed amount of time to every request so that everyone gets a fair share of his time. He will log into the system from 10am to 12am only. He wants to have separate requests queues for students and faculty. Implement a strategy for the same. The summary at the end of the session should include the total time he spent on handling queries and average query time. "

## **Solution :-**

The given problem is scheduling problem. The problem can be solved by Round Robin algorithm.

Program execution sequence:

1. Taking inputs of queries from user
2. Sorting all queries according to Arrival-Time
3. Merging all queries (initial priority to Faculty's query)
4. Applying Round-Robin algorithm on merged queries
5. Print the result

## **Steps to follow to execute the program:-**

1. Enter number of queries between 0 & 120
2. Make sure to keep value of Time-Quantum minimum
3. Enter Query Arrival Time in the format of HHMM

Example: 10:25 should be entered as 1025

4. Next Query's Arrival-Time must be less than previous Query's Completion-Time (Arrival-Time + Burst-Time)

5. Burst-Time must be entered such that  $(\text{Arrival-Time} + \text{Burst-Time}) < 120$

## **Snap-Shots Of Output :-**

```
G:\Summer Semester\Operating System\OS_Project\OS_Project_Code_AR.exe

Welcome to the OS Project made by Aryan & Rohan.

Please follow these instructions to execute the program:
1. Enter number of queries between 0 & 120
2. Make sure to keep value of TimeQuantum minimum for convinience
3. Enter Query Arrival Time in the format of HHMM
   Example: 10:25 should be entered as 1025
4. Next Query's ArrivalTime must be less than previous Query's CompletionTime (ArrivalTime + BurstTime)
5. BurstTime must be entered such that (ArrivalTime + BurstTime) < 120

Enter total number of Queries: 2

Enter Time Quantum for each query: 5

Type of Query (1 for Faculty, 2 for Student): 1

Enter Query ID: 1
Enter Query Arrival Time: 1007
Enter Burst Time: 70

Type of Query (1 for Faculty, 2 for Student): 2

Enter Query ID: 2
Enter Query Arrival Time: 1100
Enter Burst Time: 35

==> Time is in minutes for all calculations

Query ID      ArrivalTime    BurstTime      WaitingTime    TurnAroundTime  CompletionTime
1             1007 hh:mm    70 minutes    15 minutes    85 minutes      1132 hh:mm
2             1100 hh:mm    35 minutes    17 minutes    52 minutes      1152 hh:mm

Summary of Execution:

Total Time Spent on handling Queries: 105 minutes
Average TurnAround Time : 68.50 minutes
Average Waiting Time : 16.00 minutes

Program Execution Completed!

-----
Process exited after 28.74 seconds with return value 0
Press any key to continue . . .
```

Welcome to the OS Project made by Aryan & Rohan.

Please follow these instructions to execute the program:

1. Enter number of queries between 0 & 120
2. Make sure to keep value of TimeQuantum minimum for convinience
3. Enter Query Arrival Time in the format of HHMM  
Example: 10:25 should be entered as 1025
4. Next Query's ArrivalTime must be less than previous Query's CompletionTime (ArrivalTime + BurstTime)
5. BurstTime must be entered such that (ArrivalTime + BurstTime) < 120

Enter total number of Queries: 4

Enter Time Quantum for each query: 5

Type of Query (1 for Faculty, 2 for Student): 1

Enter Query ID: f1

Enter Query Arrival Time: 1000

Enter Burst Time: 40

Type of Query (1 for Faculty, 2 for Student): 1

Enter Query ID: f2

Enter Query Arrival Time: 1108

Enter Burst Time: 36

Type of Query (1 for Faculty, 2 for Student): 2

Enter Query ID: s1

Enter Query Arrival Time: 1027

Enter Burst Time: 13

Type of Query (1 for Faculty, 2 for Student): 2

Enter Query ID: s2

Enter Query Arrival Time: 1044

Enter Burst Time: 28

=> Time is in minutes for all calculations

Query ID	ArrivalTime	BurstTime	WaitingTime	TurnAroundTime	CompletionTime
f1	1000 hh:mm	40 minutes	15 minutes	55 minutes	1055 hh:mm
s1	1027 hh:mm	13 minutes	18 minutes	31 minutes	1058 hh:mm
s2	1044 hh:mm	28 minutes	24 minutes	52 minutes	1136 hh:mm
f2	1108 hh:mm	36 minutes	13 minutes	49 minutes	1157 hh:mm

Summary of Execution:

Total Time Spent on handling Queries: 117 minutes

Average TurnAround Time : 46.75 minutes

Average Waiting Time : 17.50 minutes

Program Execution Completed!

-----  
Process exited after 82.26 seconds with return value 0  
Press any key to continue . . .

Welcome to the OS Project made by Aryan & Rohan.

Please follow these instructions to execute the program:

1. Enter number of queries between 0 & 120
2. Make sure to keep value of TimeQuantum minimum for convinience
3. Enter Query Arrival Time in the format of HHMM  
Example: 10:25 should be entered as 1025
4. Next Query's ArrivalTime must be less than previous Query's CompletionTime (ArrivalTime + BurstTime)
5. BurstTime must be entered such that (ArrivalTime + BurstTime) < 120

Enter total number of Queries: 3

Enter Time Quantum for each query: 3

Type of Query (1 for Faculty, 2 for Student): 1

Enter Query ID: f1

Enter Query Arrival Time: 1030

Enter Burst Time: 15

Type of Query (1 for Faculty, 2 for Student): 2

Enter Query ID: s1

Enter Query Arrival Time: 1035

Enter Burst Time: 25

Type of Query (1 for Faculty, 2 for Student): 2

Enter Query ID: s2

Enter Query Arrival Time: 1040

Enter Burst Time: 5

=> Time is in minutes for all calculations

Query ID	ArrivalTime	BurstTime	WaitingTime	TurnAroundTime	CompletionTime
s2	1040 hh:mm	5 minutes	11 minutes	16 minutes	1056 hh:mm
f1	1030 hh:mm	15 minutes	14 minutes	29 minutes	1059 hh:mm
s1	1035 hh:mm	25 minutes	15 minutes	40 minutes	1115 hh:mm

Summary of Execution:

Total Time Spent on handling Queries: 45 minutes

Average TurnAround Time : 28.33 minutes

Average Waiting Time : 13.33 minutes

Program Execution Completed!

-----  
Process exited after 117.9 seconds with return value 0

Press any key to continue . . .

# Snap-Shots Of Code:-

```
OSProject.cpp
359 {
360     if(maximumCT < CTarr[i])
361     {
362         maximumCT = CTarr[i];
363     }
364 }
365 }
366 // Function to print Final Result of program:
367 // Time complexity = O(1)
368 void PrintResult()
369 {
370     MaxCT(); total = Mix[0].ArrivalTime;
371     printf("\n\nSummary of Execution: \n\n");
372     printf("Total Time Spent on handling Queries: %d minutes\n", maximumCT-total-1000);
373     float avgWaitTime = WaitTime * 1.0 / TotalQueries;
374     float avgTATime = TATime * 1.0 / TotalQueries;
375     printf("Average TurnAround Time : %.2f minutes\n", avgTATime);
376     printf("Average Waiting Time : %.2f minutes", avgWaitTime);
377     printf("\n\nProgram Execution Completed!\n\n");
378 }
379 // Main function:
380 // Overall Time Complexity = 2*O(n + m) + O(nLog(n)) + O(mLog(m)) + 2*O(1) = O(nLog(n)) + O(mLog(m))
381 int main() {
382     printf("\nWelcome to the OS Project made by Aryan & Rohan.\n\n");
383     "Please follow these instructions to execute the program:\n"
384     "1. Enter number of queries between 0 & 120\n"
385     "2. Make sure to keep value of TimeQuantum minimum for convenience\n"
386     "3. Enter Query Arrival Time in the format of HHMM\n"
387     "   Example: 10:25 should be entered as 1025\n"
388     "4. Next Query's ArrivalTime must be less than previous Query's CompletionTime (ArrivalTime + BurstTime)\n"
389     "5. BurstTime must be entered such that (ArrivalTime + BurstTime) < 1200\n";
390     InputsForProcess(); //Time Complexity = O(TotalQueries)
391     FacultySort(0, FacultyCount-1); // Time Complexity = O(nLog(n)); n=FacultyCount
392     StudentSort(0, StudentCount-1); // Time Complexity = O(mLog(m)); m=StudentCount
393     MergeQueries(); // Time Complexity = O(TotalQueries)
394     RoundRobin(); // Time Complexity = O(1)
395     PrintResult(); // Time Complexity = O(1)
396 }
```

```
OSProject.cpp
325     if(ATCalc>1059)
326     {
327         ATCalc += 40;
328     }
329     if(CTCalc>1059)
330     {
331         CTECalc += 40;
332     }
333     printf("\n%s\t\t%d hh:mm\t\t%d minutes\t\t%d minutes\t\t%d hh:mm",
334           Mix[i].QueryID, ATCalc, Mix[i].BurstTime,
335           total-Mix[i].ArrivalTime-Mix[i].BurstTime, total-Mix[i].ArrivalTime, CTECalc);
336     WaitTime += total - Mix[i].ArrivalTime - Mix[i].BurstTime;
337     TATime += total - Mix[i].ArrivalTime;
338     counter = 0;
339 }
340 if(i == TotalQueries - 1)
341 {
342     i = 0;
343 }
344 else if(Mix[i+1].ArrivalTime <= total)
345 {
346     i++;
347 }
348 else {
349     i = 0;
350 }
351 }
352 }
353 }
354 // Time complexity = O(1) bcoz MixCount is Limited int value
355 void MaxCT()
356 {
357     maximumCT = CTarr[0];
358     for(int i=1; i<MixCount; i++)
359     {
360         if(maximumCT < CTarr[i])
361         {
362             maximumCT = CTarr[i];
363         }
364     }
365 }
```

```
OSProject.cpp
289     else if(StudentCount == 0)
290     {
291         while(iFC != FacultyCount)
292         {
293             Mix[MixCount] = Faculty[iFC];
294             MixCount++;
295             iFC++;
296         }
297     }
298 }
299
300 // Time complexity of Round Robin = O(1)
301 void RoundRobin()
302 {
303     total = Mix[0].ArrivalTime;
304     printf("\n=> Time is in minutes for all calculations\n");
305     printf("\nQuery ID\tArrivalTime\tBurstTime\tWaitingTime\tTurnAroundTime\tCompletionTime\n");
306     for(int i = 0; i != 0;)
307     {
308         if(Mix[i].TotalTime <= TimeQuantum && Mix[i].TotalTime > 0)
309         {
310             total = total + Mix[i].TotalTime;
311             Mix[i].TotalTime = 0;
312             counter = 1;
313         }
314         else if(Mix[i].TotalTime > 0)
315         {
316             Mix[i].TotalTime -= TimeQuantum;
317             total = total + TimeQuantum;
318         }
319         if(Mix[i].TotalTime == 0 && counter == 1)
320         {
321             TQ--;
322             int ATCalc = Mix[i].ArrivalTime+1000;
323             int CTCalc = total+1000;
324             CTarr[i] = CTCalc;
325             if(ATCalc>1059)
326             {
327                 ATCalc += 40;
```

```
OSProject.cpp
253         Mix[MixCount] = Student[iSC];
254         MixCount++;
255         iSC++;
256     }
257 }
258 if(MixCount != (FacultyCount + StudentCount))
259 {
260     if(FacultyCount != iFC)
261     {
262         while(iFC != FacultyCount)
263         {
264             Mix[MixCount] = Faculty[iFC];
265             MixCount++;
266             iFC++;
267         }
268     }
269     else if(StudentCount != iSC)
270     {
271         while(iSC != StudentCount)
272         {
273             Mix[MixCount] = Student[iSC];
274             MixCount++;
275             iSC++;
276         }
277     }
278 }
279 }
280 else if(FacultyCount == 0)
281 {
282     while(iSC != StudentCount)
283     {
284         Mix[MixCount] = Student[iSC];
285         MixCount++;
286         iSC++;
287     }
288 }
289 else if(StudentCount == 0)
290 {
291     while(iFC != FacultyCount)
```



```

C:\Users\ayyan\Documents\OS Project\cpp - DevC++ 5.11
File Edit Search View Project Execute Tools Style Window Help
[Icons] (globals) OS Project.cpp
Project Classes Debug
217 Student[i+1] = Student[high];
218 Student[high] = Student[StudentCount];
219 return(i+1);
220 }
221 void StudentSort(int low, int high)
222 {
223     if(low < high)
224     {
225         int pi = Spartition(low, high);
226         StudentSort(low, pi-1);
227         StudentSort(pi+1, high);
228     }
229 }
230 // Time complexity = O(FacultyCount + StudentCount)
231 void MergeQueries()
232 {
233     int iSC=0, iFC=0;
234     if(FacultyCount !=0 && StudentCount !=0)
235     {
236         while(iSC < StudentCount && iFC < FacultyCount)
237         {
238             if(Faculty[iFC].ArrivalTime == Student[iSC].ArrivalTime)
239             {
240                 Mix[MixCount] = Faculty[iFC];
241                 MixCount++;
242                 iFC++;
243                 Mix[MixCount] = Student[iSC];
244                 MixCount++;
245                 iSC++;
246             }
247             else if(Faculty[iFC].ArrivalTime < Student[iSC].ArrivalTime) {
248                 Mix[MixCount] = Faculty[iFC];
249                 MixCount++;
250                 iFC++;
251             }
252             else if(Faculty[iFC].ArrivalTime > Student[iSC].ArrivalTime) {
253                 Mix[MixCount] = Student[iSC];
254                 MixCount++;
255                 iSC++;
256             }
257         }
258     }
259 }

```

Lines: 2 Col: 16 Sel: 0 Lines: 306 Length: 12638 Insert Done parsing in 0.14 seconds

```

C:\Users\ayyan\Documents\OS Project\cpp - DevC++ 5.11
File Edit Search View Project Execute Tools Style Window Help
[Icons] (globals) OS Project.cpp
Project Classes Debug
181 i++;
182 Faculty[FacultyCount] = Faculty[i];
183 Faculty[i] = Faculty[j];
184 Faculty[j] = Faculty[FacultyCount];
185 }
186 }
187 Faculty[FacultyCount] = Faculty[i+1];
188 Faculty[i+1] = Faculty[high];
189 Faculty[high] = Faculty[FacultyCount];
190 return(i+1);
191 }
192 void FacultySort(int low, int high)
193 {
194     if(low < high)
195     {
196         int pi = Fpartition(low, high);
197         FacultySort(low, pi-1);
198         FacultySort(pi+1, high);
199     }
200 }
201 // Time complexity of Student QuickSort = O(mLog(m)), m=no. of Student queries to sort (Limited)
202 int Spartition(int low, int high)
203 {
204     int pivot = Student[high].ArrivalTime;
205     int i = (low - 1);
206     for (int j=low; j<=high; j++)
207     {
208         if (Student[j].ArrivalTime < pivot)
209         {
210             i++;
211             Student[StudentCount] = Student[i];
212             Student[i] = Student[j];
213             Student[j] = Student[StudentCount];
214         }
215     }
216     Student[StudentCount] = Student[i+1];
217     Student[i+1] = Student[high];
218     Student[high] = Student[StudentCount];
219     return(i+1);
220 }

```

Lines: 2 Col: 16 Sel: 0 Lines: 306 Length: 12638 Insert Done parsing in 0.14 seconds

```

145         }
146     }
147     else
148     {
149         printf("\nInvalid Burst time for corresponding Arrival Time\n");
150     }
151 }
152 printf("Please enter valid Burst Time\n");
153 goto SBTTime;
154 }
155 else
156 {
157     Student[StudentCount].BurstTime = BT;
158     Burst -= BT;
159     Student[StudentCount].TotalTime = Student[StudentCount].BurstTime;
160     StudentCount++;
161 }
162 }
163 else
164 {
165     printf("\nInvalid Input. Please try again.\n");
166     goto TryQuery;
167 }
168 }
169 }
170 }
171 // Sorting Faculties and Students Queries according to Arrival Time using QuickSort algorithm:
172 // Time complexity of Faculty QuickSort = O(nLog(n)), n=no. of Faculty queries to sort (Limited)
173 int Fpartition(int low, int high)
174 {
175     int pivot = Faculty[high].ArrivalTime;
176     int i = (low - 1);
177     for (int j=low; j<=high; j++)
178     {
179         if (Faculty[j].ArrivalTime < pivot)
180         {
181             i++;
182             Faculty[FacultyCount] = Faculty[i];
183             Faculty[i] = Faculty[j];
184         }
185     }
186     Faculty[FacultyCount] = Faculty[high];
187     Faculty[high] = Faculty[i];
188     return i;
189 }
190 }
191 }
192 }
193 }
194 }
195 }
196 }
197 }
198 }
199 }
200 }
201 }
202 }
203 }
204 }
205 }
206 }
207 }
208 }
209 }
210 }
211 }
212 }
213 }
214 }
215 }
216 }
217 }
218 }
219 }
220 }
221 }
222 }
223 }
224 }
225 }
226 }
227 }
228 }
229 }
230 }
231 }
232 }
233 }
234 }
235 }
236 }
237 }
238 }
239 }
240 }
241 }
242 }
243 }
244 }
245 }
246 }
247 }
248 }
249 }
250 }
251 }
252 }
253 }
254 }
255 }
256 }
257 }
258 }
259 }
260 }
261 }
262 }
263 }
264 }
265 }
266 }
267 }
268 }
269 }
270 }
271 }
272 }
273 }
274 }
275 }
276 }
277 }
278 }
279 }
280 }
281 }
282 }
283 }
284 }
285 }
286 }
287 }
288 }
289 }
290 }
291 }
292 }
293 }
294 }
295 }
296 }
297 }
298 }
299 }
300 }
301 }
302 }
303 }
304 }
305 }
306 }
307 }
308 }
309 }
310 }
311 }
312 }
313 }
314 }
315 }
316 }
317 }
318 }
319 }
320 }
321 }
322 }
323 }
324 }
325 }
326 }
327 }
328 }
329 }
330 }
331 }
332 }
333 }
334 }
335 }
336 }
337 }
338 }
339 }
340 }
341 }
342 }
343 }
344 }
345 }
346 }
347 }
348 }
349 }
350 }
351 }
352 }
353 }
354 }
355 }
356 }
357 }
358 }
359 }
360 }
361 }
362 }
363 }
364 }
365 }
366 }
367 }
368 }
369 }
370 }
371 }
372 }
373 }
374 }
375 }
376 }
377 }
378 }
379 }
380 }
381 }
382 }
383 }
384 }
385 }
386 }
387 }
388 }
389 }
390 }
391 }
392 }
393 }
394 }
395 }
396 }
397 }
398 }
399 }
400 }
401 }
402 }
403 }
404 }
405 }
406 }
407 }
408 }
409 }
410 }
411 }
412 }
413 }
414 }
415 }
416 }
417 }
418 }
419 }
420 }
421 }
422 }
423 }
424 }
425 }
426 }
427 }
428 }
429 }
430 }
431 }
432 }
433 }
434 }
435 }
436 }
437 }
438 }
439 }
440 }
441 }
442 }
443 }
444 }
445 }
446 }
447 }
448 }
449 }
450 }
451 }
452 }
453 }
454 }
455 }
456 }
457 }
458 }
459 }
460 }
461 }
462 }
463 }
464 }
465 }
466 }
467 }
468 }
469 }
470 }
471 }
472 }
473 }
474 }
475 }
476 }
477 }
478 }
479 }
480 }
481 }
482 }
483 }
484 }
485 }
486 }
487 }
488 }
489 }
490 }
491 }
492 }
493 }
494 }
495 }
496 }
497 }
498 }
499 }
500 }
501 }
502 }
503 }
504 }
505 }
506 }
507 }
508 }
509 }
510 }
511 }
512 }
513 }
514 }
515 }
516 }
517 }
518 }
519 }
520 }
521 }
522 }
523 }
524 }
525 }
526 }
527 }
528 }
529 }
530 }
531 }
532 }
533 }
534 }
535 }
536 }
537 }
538 }
539 }
540 }
541 }
542 }
543 }
544 }
545 }
546 }
547 }
548 }
549 }
550 }
551 }
552 }
553 }
554 }
555 }
556 }
557 }
558 }
559 }
560 }
561 }
562 }
563 }
564 }
565 }
566 }
567 }
568 }
569 }
570 }
571 }
572 }
573 }
574 }
575 }
576 }
577 }
578 }
579 }
580 }
581 }
582 }
583 }
584 }
585 }
586 }
587 }
588 }
589 }
590 }
591 }
592 }
593 }
594 }
595 }
596 }
597 }
598 }
599 }
600 }
601 }
602 }
603 }
604 }
605 }
606 }
607 }
608 }
609 }
610 }
611 }
612 }
613 }
614 }
615 }
616 }
617 }
618 }
619 }
620 }
621 }
622 }
623 }
624 }
625 }
626 }
627 }
628 }
629 }
630 }
631 }
632 }
633 }
634 }
635 }
636 }
637 }
638 }
639 }
640 }
641 }
642 }
643 }
644 }
645 }
646 }
647 }
648 }
649 }
650 }
651 }
652 }
653 }
654 }
655 }
656 }
657 }
658 }
659 }
660 }
661 }
662 }
663 }
664 }
665 }
666 }
667 }
668 }
669 }
670 }
671 }
672 }
673 }
674 }
675 }
676 }
677 }
678 }
679 }
680 }
681 }
682 }
683 }
684 }
685 }
686 }
687 }
688 }
689 }
690 }
691 }
692 }
693 }
694 }
695 }
696 }
697 }
698 }
699 }
700 }
701 }
702 }
703 }
704 }
705 }
706 }
707 }
708 }
709 }
710 }
711 }
712 }
713 }
714 }
715 }
716 }
717 }
718 }
719 }
720 }
721 }
722 }
723 }
724 }
725 }
726 }
727 }
728 }
729 }
730 }
731 }
732 }
733 }
734 }
735 }
736 }
737 }
738 }
739 }
740 }
741 }
742 }
743 }
744 }
745 }
746 }
747 }
748 }
749 }
750 }
751 }
752 }
753 }
754 }
755 }
756 }
757 }
758 }
759 }
760 }
761 }
762 }
763 }
764 }
765 }
766 }
767 }
768 }
769 }
770 }
771 }
772 }
773 }
774 }
775 }
776 }
777 }
778 }
779 }
780 }
781 }
782 }
783 }
784 }
785 }
786 }
787 }
788 }
789 }
790 }
791 }
792 }
793 }
794 }
795 }
796 }
797 }
798 }
799 }
800 }
801 }
802 }
803 }
804 }
805 }
806 }
807 }
808 }
809 }
810 }
811 }
812 }
813 }
814 }
815 }
816 }
817 }
818 }
819 }
820 }
821 }
822 }
823 }
824 }
825 }
826 }
827 }
828 }
829 }
830 }
831 }
832 }
833 }
834 }
835 }
836 }
837 }
838 }
839 }
840 }
841 }
842 }
843 }
844 }
845 }
846 }
847 }
848 }
849 }
850 }
851 }
852 }
853 }
854 }
855 }
856 }
857 }
858 }
859 }
860 }
861 }
862 }
863 }
864 }
865 }
866 }
867 }
868 }
869 }
870 }
871 }
872 }
873 }
874 }
875 }
876 }
877 }
878 }
879 }
880 }
881 }
882 }
883 }
884 }
885 }
886 }
887 }
888 }
889 }
890 }
891 }
892 }
893 }
894 }
895 }
896 }
897 }
898 }
899 }
900 }
901 }
902 }
903 }
904 }
905 }
906 }
907 }
908 }
909 }
910 }
911 }
912 }
913 }
914 }
915 }
916 }
917 }
918 }
919 }
920 }
921 }
922 }
923 }
924 }
925 }
926 }
927 }
928 }
929 }
930 }
931 }
932 }
933 }
934 }
935 }
936 }
937 }
938 }
939 }
940 }
941 }
942 }
943 }
944 }
945 }
946 }
947 }
948 }
949 }
950 }
951 }
952 }
953 }
954 }
955 }
956 }
957 }
958 }
959 }
960 }
961 }
962 }
963 }
964 }
965 }
966 }
967 }
968 }
969 }
970 }
971 }
972 }
973 }
974 }
975 }
976 }
977 }
978 }
979 }
980 }
981 }
982 }
983 }
984 }
985 }
986 }
987 }
988 }
989 }
990 }
991 }
992 }
993 }
994 }
995 }
996 }
997 }
998 }
999 }
1000 }

```

```

109     goto STime;
110 }
111 else
112 {
113     if (AT>=1000 && AT<1100)
114     {
115         Student[StudentCount].ArrivalTime = AT-1000;
116     }
117     else {
118         Student[StudentCount].ArrivalTime = AT-1040;
119     }
120 }
121 SBTTime:
122 printf("Enter Burst Time: ");
123 scanf("%d", &BT);
124 if(Burst - BT < 0 || BT <= 0 || Student[StudentCount].ArrivalTime + BT >= 120)
125 {
126     if(BT<=0)
127     {
128         printf("\nBurst Time cannot be less than 0\n");
129     }
130     else {
131         if (Burst-BT<=0)
132         {
133             int choice;
134             printf("\nKaran won't have enough time to handle this Query because of high BurstTime."
135                 "\nWant to change BurstTime? (1 : Yes; Else : No) ");
136             scanf("%d", &choice);
137             if(choice==1)
138             {
139                 goto FBTTime;
140             }
141             else
142             {
143                 printf("\nOK. This query's all data will be lost\n");
144                 goto TryQuery;
145             }
146         }
147     }
148 }
149 }
150 }
151 }
152 }
153 }
154 }
155 }
156 }
157 }
158 }
159 }
160 }
161 }
162 }
163 }
164 }
165 }
166 }
167 }
168 }
169 }
170 }
171 }
172 }
173 }
174 }
175 }
176 }
177 }
178 }
179 }
180 }
181 }
182 }
183 }
184 }
185 }
186 }
187 }
188 }
189 }
190 }
191 }
192 }
193 }
194 }
195 }
196 }
197 }
198 }
199 }
200 }
201 }
202 }
203 }
204 }
205 }
206 }
207 }
208 }
209 }
210 }
211 }
212 }
213 }
214 }
215 }
216 }
217 }
218 }
219 }
220 }
221 }
222 }
223 }
224 }
225 }
226 }
227 }
228 }
229 }
230 }
231 }
232 }
233 }
234 }
235 }
236 }
237 }
238 }
239 }
240 }
241 }
242 }
243 }
244 }
245 }
246 }
247 }
248 }
249 }
250 }
251 }
252 }
253 }
254 }
255 }
256 }
257 }
258 }
259 }
260 }
261 }
262 }
263 }
264 }
265 }
266 }
267 }
268 }
269 }
270 }
271 }
272 }
273 }
274 }
275 }
276 }
277 }
278 }
279 }
280 }
281 }
282 }
283 }
284 }
285 }
286 }
287 }
288 }
289 }
290 }
291 }
292 }
293 }
294 }
295 }
296 }
297 }
298 }
299 }
300 }
301 }
302 }
303 }
304 }
305 }
306 }
307 }
308 }
309 }
310 }
311 }
312 }
313 }
314 }
315 }
316 }
317 }
318 }
319 }
320 }
321 }
322 }
323 }
324 }
325 }
326 }
327 }
328 }
329 }
330 }
331 }
332 }
333 }
334 }
335 }
336 }
337 }
338 }
339 }
340 }
341 }
342 }
343 }
344 }
345 }
346 }
347 }
348 }
349 }
350 }
351 }
352 }
353 }
354 }
355 }
356 }
357 }
358 }
359 }
360 }
361 }
362 }
363 }
364 }
365 }
366 }
367 }
368 }
369 }
370 }
371 }
372 }
373 }
374 }
375 }
376 }
377 }
378 }
379 }
380 }
381 }
382 }
383 }
384 }
385 }
386 }
387 }
388 }
389 }
390 }
391 }
392 }
393 }
394 }
395 }
396 }
397 }
398 }
399 }
400 }
401 }
402 }
403 }
404 }
405 }
406 }
407 }
408 }
409 }
410 }
411 }
412 }
413 }
414 }
415 }
416 }
417 }
418 }
419 }
420 }
421 }
422 }
423 }
424 }
425 }
426 }
427 }
428 }
429 }
430 }
431 }
432 }
433 }
434 }
435 }
436 }
437 }
438 }
439 }
440 }
441 }
442 }
443 }
444 }
445 }
446 }
447 }
448 }
449 }
450 }
451 }
452 }
453 }
454 }
455 }
456 }
457 }
458 }
459 }
460 }
461 }
462 }
463 }
464 }
465 }
466 }
467 }
468 }
469 }
470 }
471 }
472 }
473 }
474 }
475 }
476 }
477 }
478 }
479 }
480 }
481 }
482 }
483 }
484 }
485 }
486 }
487 }
488 }
489 }
490 }
491 }
492 }
493 }
494 }
495 }
496 }
497 }
498 }
499 }
500 }
501 }
502 }
503 }
504 }
505 }
506 }
507 }
508 }
509 }
510 }
511 }
512 }
513 }
514 }
515 }
516 }
517 }
518 }
519 }
520 }
521 }
522 }
523 }
524 }
525 }
526 }
527 }
528 }
529 }
530 }
531 }
532 }
533 }
534 }
535 }
536 }
537 }
538 }
539 }
540 }
541 }
542 }
543 }
544 }
545 }
546 }
547 }
548 }
549 }
550 }
551 }
552 }
553 }
554 }
555 }
556 }
557 }
558 }
559 }
560 }
561 }
562 }
563 }
564 }
565 }
566 }
567 }
568 }
569 }
570 }
571 }
572 }
573 }
574 }
575 }
576 }
577 }
578 }
579 }
580 }
581 }
582 }
583 }
584 }
585 }
586 }
587 }
588 }
589 }
590 }
591 }
592 }
593 }
594 }
595 }
596 }
597 }
598 }
599 }
600 }
601 }
602 }
603 }
604 }
605 }
606 }
607 }
608 }
609 }
610 }
611 }
612 }
613 }
614 }
615 }
616 }
617 }
618 }
619 }
620 }
621 }
622 }
623 }
624 }
625 }
626 }
627 }
628 }
629 }
630 }
631 }
632 }
633 }
634 }
635 }
636 }
637 }
638 }
639 }
640 }
641 }
642 }
643 }
644 }
645 }
646 }
647 }
648 }
649 }
650 }
651 }
652 }
653 }
654 }
655 }
656 }
657 }
658 }
659 }
660 }
661 }
662 }
663 }
664 }
665 }
666 }
667 }
668 }
669 }
670 }
671 }
672 }
673 }
674 }
675 }
676 }
677 }
678 }
679 }
680 }
681 }
682 }
683 }
684 }
685 }
686 }
687 }
688 }
689 }
690 }
691 }
692 }
693 }
694 }
695 }
696 }
697 }
698 }
699 }
700 }
701 }
702 }
703 }
704 }
705 }
706 }
707 }
708 }
709 }
710 }
711 }
712 }
713 }
714 }
715 }
716 }
717 }
718 }
719 }
720 }
721 }
722 }
723 }
724 }
725 }
726 }
727 }
728 }
729 }
730 }
731 }
732 }
733 }
734 }
735 }
736 }
737 }
738 }
739 }
740 }
741 }
742 }
743 }
744 }
745 }
746 }
747 }
748 }
749 }
750 }
751 }
752 }
753 }
754 }
755 }
756 }
757 }
758 }
759 }
760 }
761 }
762 }
763 }
764 }
765 }
766 }
767 }
768 }
769 }
770 }
771 }
772 }
773 }
774 }
775 }
776 }
777 }
778 }
779 }
780 }
781 }
782 }
783 }
784 }
785 }
786 }
787 }
788 }
789 }
790 }
791 }
792 }
793 }
794 }
795 }
796 }
797 }
798 }
799 }
800 }
801 }
802 }
803 }
804 }
805 }
806 }
807 }
808 }
809 }
810 }
811 }
812 }
813 }
814 }
815 }
816 }
817 }
818 }
819 }
820 }
821 }
822 }
823 }
824 }
825 }
826 }
827 }
828 }
829 }
830 }
831 }
832 }
833 }
834 }
835 }
836 }
837 }
838 }
839 }
840 }
841 }
842 }
843 }
844 }
845 }
846 }
847 }
848 }
849 }
850 }
851 }
852 }
853 }
854 }
855 }
856 }
857 }
858 }
859 }
860 }
861 }
862 }
863 }
864 }
865 }
866 }
867 }
868 }
869 }
870 }
871 }
872 }
873 }
874 }
875 }
876 }
877 }
878 }
879 }
880 }
881 }
882 }
883 }
884 }
885 }
886 }
887 }
888 }
889 }
890 }
891 }
892 }
893 }
894 }
895 }
896 }
897 }
898 }
899 }
900 }
901 }
902 }
903 }
904 }
905 }
906 }
907 }
908 }
909 }
910 }
911 }
912 }
913 }
914 }
915 }
916 }
917 }
918 }
919 }
920 }
921 }
922 }
923 }
924 }
925 }
926 }
927 }
928 }
929 }
930 }
931 }
932 }
933 }
934 }
935 }
936 }
937 }
938 }
939 }
940 }
941 }
942 }
943 }
944 }
945 }
946 }
947 }
948 }
949 }
950 }
951 }
952 }
953 }
954 }
955 }
956 }
957 }
958 }
959 }
960 }
961 }
962 }
963 }
964 }
965 }
966 }
967 }
968 }
969 }
970 }
971 }
972 }
973 }
974 }
975 }
976 }
977 }
978 }
979 }
980 }
981 }
982 }
983 }
984 }
985 }
986 }
987 }
988 }
989 }
990 }
991 }
992 }
993 }
994 }
995 }
996 }
997 }
998 }
999 }
1000 }

```

```
OSProject.cpp
73 scanf("%d", &choice);
74 if(choice==1) { goto FBTime; }
75 else
76 {
77     printf("\nOK. This query's all data will be lost\n");
78     goto TryQuery;
79 }
80
81 else
82 {
83     printf("\nInvalid Burst time for corresponding Arrival Time\n");
84 }
85
86 printf("Please enter valid Burst Time\n");
87 goto FBTime;
88
89 else
90 {
91     Faculty[FacultyCount].BurstTime = BT;
92 }
93 Burst -= BT;
94 Faculty[FacultyCount].TotalTime = Faculty[FacultyCount].BurstTime;
95 FacultyCount++;
96 }
97
98 // For Student
99 else if(QueryType == 2)
100 {
101     printf("\nEnter Query ID: ");
102     scanf("%s", &Student[StudentCount].QueryID[0]);
103     STime:
104     printf("Enter Query Arrival Time: ");
105     scanf("%d", &AT);
106     if(AT<1000 || AT>1200 || (AT<1100 && AT>1060) || (AT<1200 && AT>1160))
107     {
108         printf("\nEnter valid Time!\n");
109         goto STime;
110     }
111     else
```

```
OSProject.cpp
37 printf("\nEnter Query ID: ");
38 scanf("%s", &Faculty[FacultyCount].QueryID[0]);
39 FTime:
40 printf("Enter Query Arrival Time: ");
41 scanf("%d", &AT);
42 if(AT<1000 || AT>1200 || (AT<1100 && AT>1059) || (AT<1200 && AT>1159))
43 {
44     printf("\nEnter Correct Time!\n");
45     goto FTime;
46 }
47 else
48 {
49     if (AT>=1000 && AT<1100)
50     {
51         Faculty[FacultyCount].ArrivalTime = AT-1000;
52     }
53     else {
54         Faculty[FacultyCount].ArrivalTime = AT-1040;
55     }
56 }
57 FBTime:
58 printf("Enter Burst Time: ");
59 scanf("%d", &BT);
60 if(Burst - BT < 0 || BT <= 0 || Faculty[FacultyCount].ArrivalTime + BT >= 120)
61 {
62     if(BT<=0)
63     {
64         printf("\nBurst Time cannot be less than 0\n");
65     }
66     else
67     {
68         if (Burst-BT<=0)
69         {
70             int choice;
71             printf("\nKaran will not have enough time to handle this Query because of high BurstTime."
72                 "\nWant to change BurstTime? (1 : Yes; Else : No) ");
73             scanf("%d", &choice);
74             if(choice==1) { goto FBTime; }
75         }
76     }
77 }
```

```
OSProject.cpp
359 {
360     if(maximumCT < CTarr[i])
361     {
362         maximumCT = CTarr[i];
363     }
364 }
365 // Function to print Final Result of program:
366 // Time complexity = O(1)
367 void PrintResult()
368 {
369     MaxCT(); total = Mix[0].ArrivalTime;
370     printf("\n\nSummary of Execution: \n\n");
371     printf("Total Time Spent on handling Queries: %d minutes\n", maximumCT-total-1000);
372     float avgWaitTime = WaitTime * 1.0 / TotalQueries;
373     float avgTATime = TATime * 1.0 / TotalQueries;
374     printf("Average TurnAround Time : %.2f minutes\n", avgTATime);
375     printf("Average Waiting Time : %.2f minutes", avgWaitTime);
376     printf("\n\nProgram Execution Completed!\n\n");
377 }
378 // Main function:
379 // Overall Time Complexity = 2*O(n + m) + O(nLog(n)) + O(mLog(m)) + 2*O(1) = O(nLog(n)) + O(mLog(m))
380 int main() {
381     printf("\nWelcome to the OS Project made by Aryan & Rohan.\n\n");
382     "Please follow these instructions to execute the program:\n";
383     "1. Enter number of queries between 0 & 120\n";
384     "2. Make sure to keep value of TimeQuantum minimum for convenience\n";
385     "3. Enter Query Arrival Time in the format of HHMM\n";
386     "    Example: 10:25 should be entered as 1025\n";
387     "4. Next Query's ArrivalTime must be less than previous Query's CompletionTime (ArrivalTime + BurstTime)\n";
388     "5. BurstTime must be entered such that (ArrivalTime + BurstTime) < 120\n";
389     InputsForProcess(); //Time Complexity = O(TotalQueries)
390     FacultySort(0, FacultyCount-1); // Time Complexity = O(nLog(n)); n=FacultyCount
391     StudentSort(0, StudentCount-1); // Time Complexity = O(mLog(m)); m=StudentCount
392     MergeQueries(); // Time Complexity = O(TotalQueries)
393     RoundRobin(); // Time Complexity = O(1)
394     PrintResult(); // Time Complexity = O(1)
395 }
396 }
```

Thank You 😊