Lab Assignment 01

Course Title :- Operating System Lab

Course Code: CSE – 3202

Submitted to :-

Md. Zahidur Rahman

Lecturer

Department of CSE,

Comilla University, Cumilla

Submitted by :-

Name: Md Gulam Sarwar Remon

ID – 12108050

Session-2020-21

Department of CSE,

Comilla University, Cumilla

Department of CSE

Comilla University

Date of submission:- 13.04.2025

Experiment No.: 1

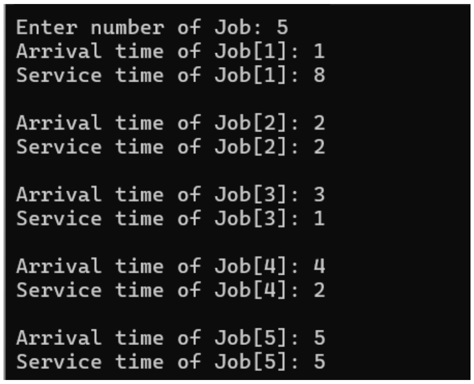
Experiment Name: Write a C program to simulate the FCFS CPU scheduling algorithms to find turnaround time and waiting time for a problem.

Source Code:

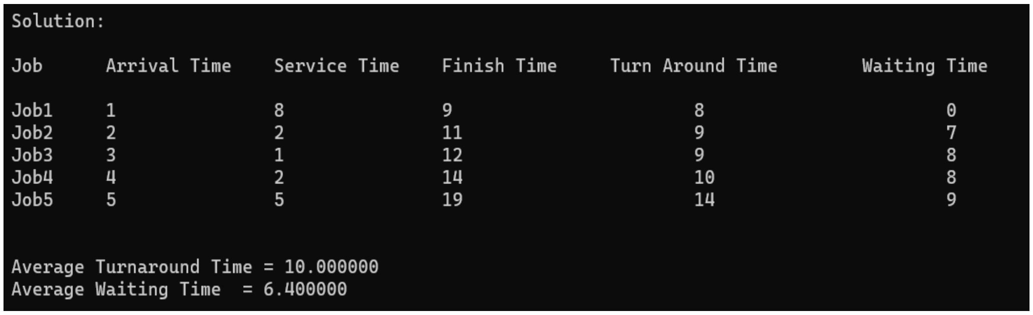
```
#include<stdio.h>

int main(){

   int at[10]={0}, st[10]={0}, ft[10]={0},tat[10]={0},wt[10]={0};
   int n,i,k,j,sum=0;
   float totalTAT=0,totalWT=0;

   printf("Enter number of Job: ");
   scanf("%d",&n);

   //Input Arrival time and Service Time for each job
   for(i=0;i<n;i++)
   {
      printf("Arrival time of Job[%d]: ",i+1);
      scanf("%d",&at[i]);

      printf("Service time of Job[%d]: ",i+1);
      scanf("%d",&st[i]);

      printf("\n");
   }

   //Calculating Finishing Time (Gantt Chart)
   sum=sum+at[0];   //for 1st case i.e. 1st job, sum=1
```

```
    for(j=0;j<n;j++)
    {
        ft[j] = sum + st[j]; //here, 'sum' is considered as 'starting time'
        sum   = ft[j]; }

    //Calculating Waiting and Turnaround Time
    for(k=0;k<n;k++)
    {
        tat[k]=ft[k]-at[k];
        wt[k]=tat[k]-st[k];
        totalTAT+=tat[k];
        totalWT+=wt[k];
    }
    printf("Solution: \n\n");
    printf("Job\t Arrival Time\t Service Time\t Finish Time\t Turn Around Time\t Waiting
   Time\t\n\n");

    for(i=0;i<n;i++)
    {  printf("Job%d\t %d\t\t %d\t\t\ %d\t\t\t %d\t\t\t %d\n",i+1,at[i],st[i],ft[i],tat[i],wt[i]);
    }

    printf("\n\nAverage Turnaround Time = %f\n",totalTAT/n);
    printf("Average Waiting Time  = %f\n\n",totalWT/n);

    return 0;
}
```

Sample Input and Output:



```
Enter number of Job: 5
Arrival time of Job[1]: 1
Service time of Job[1]: 8

Arrival time of Job[2]: 2
Service time of Job[2]: 2

Arrival time of Job[3]: 3
Service time of Job[3]: 1

Arrival time of Job[4]: 4
Service time of Job[4]: 2

Arrival time of Job[5]: 5
Service time of Job[5]: 5
```



```
Solution:

Job      Arrival Time    Service Time    Finish Time     Turn Around Time        Waiting Time

Job1     1               8               9                       8                        0
Job2     2               2               11                      9                        7
Job3     3               1               12                      9                        8
Job4     4               2               14                      10                       8
Job5     5               5               19                      14                       9


Average Turnaround Time = 10.000000
Average Waiting Time  = 6.400000
```

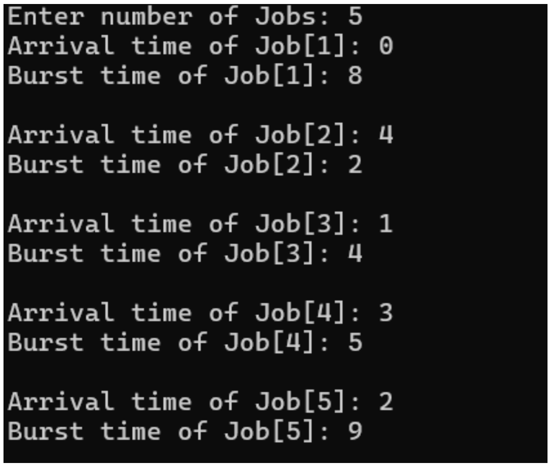
Experiment No.:2

Experiment Name: Write a C program to simulate the SJF CPU scheduling algorithms to find turnaround time and waiting time for a problem.

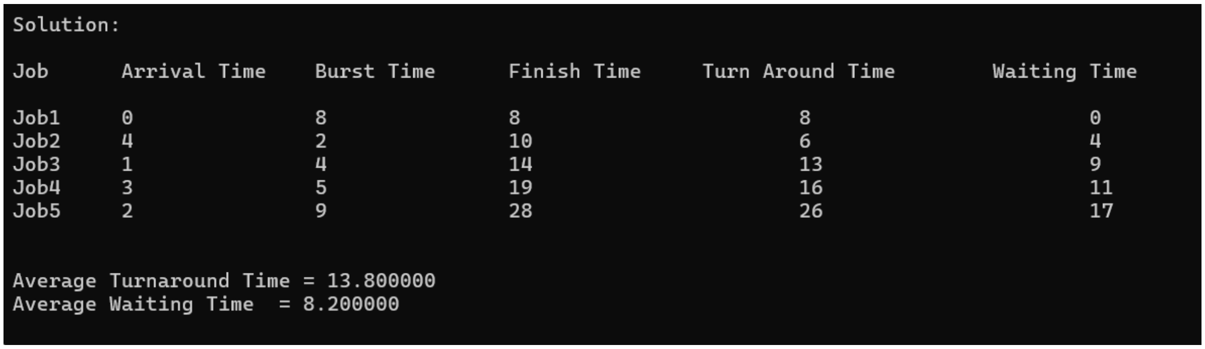
Source Code:

```
#include<stdio.h>

#include<limits.h>

int main(){

    int at[10]={0}, bt[10]={0}, ft[10]={0},tat[10]={0},wt[10]={0};

    int n,i,k,j,sum=0, completed[10]={0};

    float totalTAT=0,totalWT=0;

    printf("Enter number of Jobs: ");

    scanf("%d",&n);

    //Input Arrival time and Burst Time for each job

    for(i=0;i<n;i++)

    {

        printf("Arrival time of Job[%d]: ",i+1);

        scanf("%d",&at[i]);

        printf("Burst time of Job[%d]: ",i+1);

        scanf("%d",&bt[i]);

        printf("\n");

    }

    //Calculating Finishing Time (Gantt Chart)

    int current_time = 0, completed_jobs = 0;

    while (completed_jobs < n) {

        int shortest_job = -1, shortest_burst = INT_MAX;

        for (i = 0; i < n; i++) {
            if (at[i] <= current_time && completed[i] == 0 && bt[i] < shortest_burst) {
                shortest_job = i;
                shortest_burst = bt[i];
            }
        }
        if (shortest_job == -1) current_time++;
        else {
            ft[shortest_job] = current_time + bt[shortest_job];
            tat[shortest_job] = ft[shortest_job] - at[shortest_job];
            wt[shortest_job] = tat[shortest_job] - bt[shortest_job];
            completed[shortest_job] = 1;
            completed_jobs++;
            current_time = ft[shortest_job];
        }
    }
    //Calculating Waiting and Turnaround Time
    for(k=0;k<n;k++)
    {
        totalTAT+=tat[k];
        totalWT+=wt[k];
    }
    printf("Solution: \n\n");
    printf("Job\t Arrival Time\t Burst Time\t Finish Time\t Turn Around Time\t Waiting
Time\t\n\n");
    for(i=0;i<n;i++)
    {
        printf("Job%d\t %d\t\t %d\t\t\ %d\t\t\t %d\t\t\t %d\n",i+1,at[i],bt[i],ft[i],tat[i],wt[i]);
    }
    printf("\n\nAverage Turnaround Time = %f\n",totalTAT/n);
    printf("Average Waiting Time  = %f\n\n",totalWT/n);
    return 0;
}
```

Sample Input and Output:



```
Enter number of Jobs: 5
Arrival time of Job[1]: 0
Burst time of Job[1]: 8

Arrival time of Job[2]: 4
Burst time of Job[2]: 2

Arrival time of Job[3]: 1
Burst time of Job[3]: 4

Arrival time of Job[4]: 3
Burst time of Job[4]: 5

Arrival time of Job[5]: 2
Burst time of Job[5]: 9
```



```
Solution:

Job     Arrival Time    Burst Time      Finish Time     Turn Around Time        Waiting Time

Job1    0               8               8                       8                       0
Job2    4               2               10                      6                       4
Job3    1               4               14                      13                      9
Job4    3               5               19                      16                      11
Job5    2               9               28                      26                      17


Average Turnaround Time = 13.800000
Average Waiting Time  = 8.200000
```

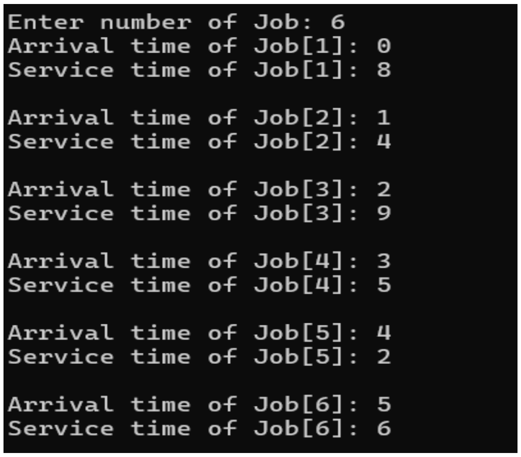
Experiment No.:3

Experiment Name: Write a C program to simulate the SRTF CPU scheduling algorithms to find turnaround time and waiting time for a problem.

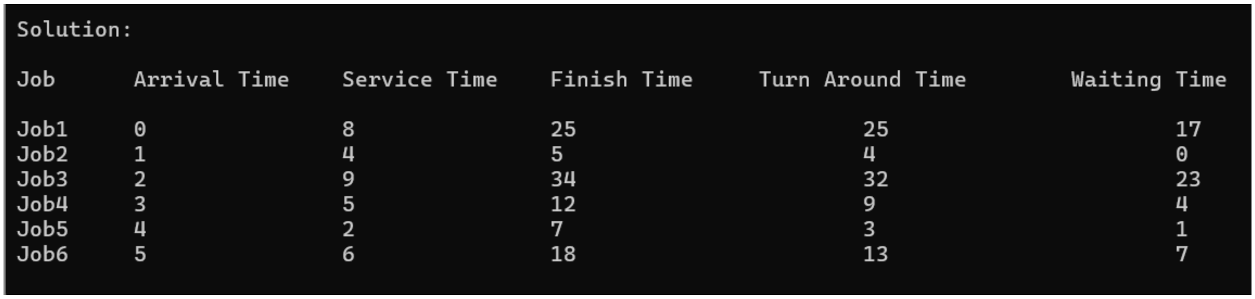
Source Code:

```
#include<stdio.h>
#include<limits.h>
int main(){
   int at[10]={0}, st[10]={0}, ft[10]={0},tat[10]={0},wt[10]={0}, remaining_time[10]={0};
   int n,i,k,j,sum=0, completed_jobs = 0, current_time = 0;
   float totalTAT=0,totalWT=0;
   printf("Enter number of Job: ");
   scanf("%d",&n);
   //Input Arrival time and Service Time for each job
   for(i=0;i<n;i++)
   {
      printf("Arrival time of Job[%d]: ",i+1);
      scanf("%d",&at[i]);
      printf("Service time of Job[%d]: ",i+1);
      scanf("%d",&st[i]);
      remaining_time[i] = st[i];
      printf("\n");
   }
   //Calculating Finishing Time (Gantt Chart)
   while (completed_jobs < n) {
      int shortest_job = -1, shortest_remaining = INT_MAX;
      for (i = 0; i < n; i++) {
            if (at[i] <= current_time && remaining_time[i] > 0 && remaining_time[i] < shortest_remaining) {
                shortest_job = i;
                shortest_remaining = remaining_time[i];
            }
        }
        if (shortest_job == -1) current_time++;
        else {
            remaining_time[shortest_job]--;
            current_time++;
            if (remaining_time[shortest_job] == 0) {
                ft[shortest_job] = current_time;
                tat[shortest_job] = ft[shortest_job] - at[shortest_job];
                wt[shortest_job] = tat[shortest_job] - st[shortest_job];
                completed_jobs++;
            }
        }
    }

    //Calculating Waiting and Turnaround Time
    for(k=0;k<n;k++)
    {
        totalTAT+=tat[k];
        totalWT+=wt[k];
    }
    printf("Solution: \n\n");
    printf("Job\t Arrival Time\t Service Time\t Finish Time\t Turn Around Time\t Waiting
Time\t\n\n");
    for(i=0;i<n;i++)
    {
        printf("Job%d\t %d\t\t %d\t\t\ %d\t\t\t %d\t\t\t %d\n",i+1,at[i],st[i],ft[i],tat[i],wt[i]);
    }
    printf("\n\nAverage Turnaround Time = %f\n",totalTAT/n);
    printf("Average Waiting Time  = %f\n\n",totalWT/n);
    return 0;
}
```

Sample Input and Output:



```
Enter number of Job: 6
Arrival time of Job[1]: 0
Service time of Job[1]: 8

Arrival time of Job[2]: 1
Service time of Job[2]: 4

Arrival time of Job[3]: 2
Service time of Job[3]: 9

Arrival time of Job[4]: 3
Service time of Job[4]: 5

Arrival time of Job[5]: 4
Service time of Job[5]: 2

Arrival time of Job[6]: 5
Service time of Job[6]: 6
```



```
Solution:

Job     Arrival Time    Service Time    Finish Time     Turn Around Time        Waiting Time

Job1    0               8               25                  25                      17
Job2    1               4               5                   4                       0
Job3    2               9               34                  32                      23
Job4    3               5               12                  9                       4
Job5    4               2               7                   3                       1
Job6    5               6               18                  13                      7
```

Experiment No: 4

Experiment Name: Write a C program to simulate the Round Robin CPU scheduling algorithms to find turnaround time and waiting time for a problem.

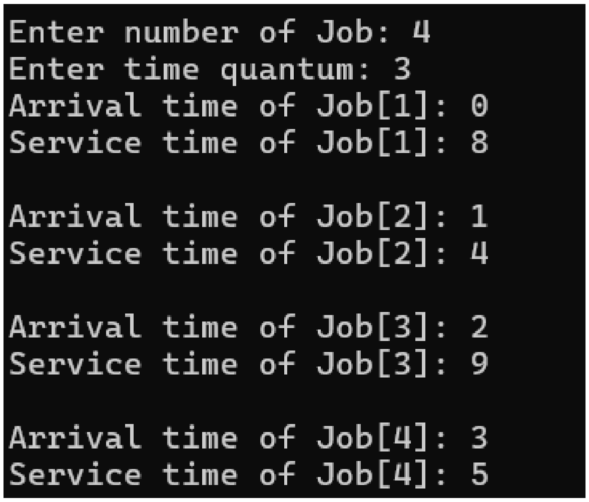
Source Code:

```
#include<stdio.h>
int main(){
    int at[10]={0}, st[10]={0}, ft[10]={0},tat[10]={0},wt[10]={0}, remaining_time[10]={0};
    int n,i,k,j,sum=0, time_quantum;
    float totalTAT=0,totalWT=0;
    printf("Enter number of Job: ");
    scanf("%d",&n);
    printf("Enter time quantum: ");
    scanf("%d", &time_quantum);
    //Input Arrival time and Service Time for each job
    for(i=0;i<n;i++)
    {
        printf("Arrival time of Job[%d]: ",i+1);
        scanf("%d",&at[i]);

        printf("Service time of Job[%d]: ",i+1);
        scanf("%d",&st[i]);
        remaining_time[i] = st[i];
        printf("\n");
    }
    //Calculating Finishing Time (Gantt Chart)
    int current_time = 0;
```

```
while (1) {
    int done = 1;
    for (i = 0; i < n; i++) {
        if (remaining_time[i] > 0) {
            done = 0;
            if (remaining_time[i] > time_quantum) {
                current_time += time_quantum;
                remaining_time[i] -= time_quantum;
            } else {
                current_time += remaining_time[i];
                ft[i] = current_time;
                tat[i] = ft[i] - at[i];
                wt[i] = tat[i] - st[i];
                remaining_time[i] = 0;
            }
        }
    }
    if (done) break;
}
//Calculating Waiting and Turnaround Time
for(k=0;k<n;k++)
{
    totalTAT+=tat[k];
    totalWT+=wt[k];
}
printf("Solution: \n\n");
```

```
    printf("Job\t Arrival Time\t Service Time\t Finish Time\t Turn Around Time\t Waiting
Time\t\n\n");
    for(i=0;i<n;i++)
    {
        printf("Job%d\t %d\t\t %d\t\t\ %d\t\t\t %d\t\t\t %d\n",i+1,at[i],st[i],ft[i],tat[i],wt[i]);
    }
    printf("\n\nAverage Turnaround Time = %f\n",totalTAT/n);
    printf("Average Waiting Time  = %f\n\n",totalWT/n);
    return 0;
}
```

Sample Input and Output:



```
Enter number of Job: 4
Enter time quantum: 3
Arrival time of Job[1]: 0
Service time of Job[1]: 8

Arrival time of Job[2]: 1
Service time of Job[2]: 4

Arrival time of Job[3]: 2
Service time of Job[3]: 9

Arrival time of Job[4]: 3
Service time of Job[4]: 5
```

Experiment No.:5

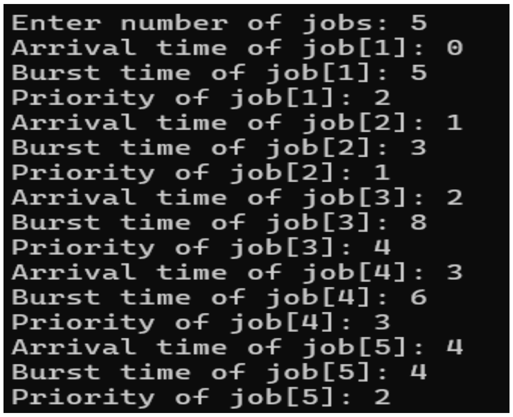
Experiment Name: Write a C program to simulate the Priority CPU scheduling algorithms to find turnaround time and waiting time for a problem.

Source Code:

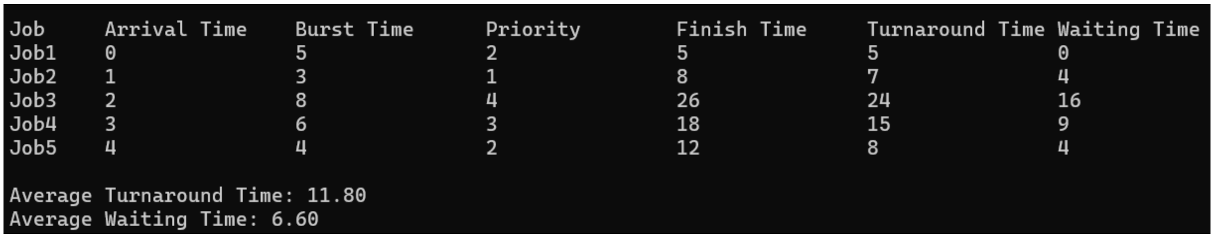
```
#include <stdio.h>
#include <limits.h>
int main() {
    int at[10], bt[10], pt[10], ft[10], tat[10], wt[10], completed[10];
    int n, i, j, current_time, completed_jobs, highest_priority_job, highest_priority;
    float totalTAT = 0, totalWT = 0;
    printf("Enter number of jobs: ");
    scanf("%d", &n);
    // Input arrival time, burst time, and priority for each job
    for (i = 0; i < n; i++) {
        printf("Arrival time of job[%d]: ", i + 1);
        scanf("%d", &at[i]);
        printf("Burst time of job[%d]: ", i + 1);
        scanf("%d", &bt[i]);
        printf("Priority of job[%d]: ", i + 1);
        scanf("%d", &pt[i]);
        completed[i] = 0; // Initially, all jobs are marked as incomplete
    }
    current_time = 0;
    completed_jobs = 0;
    while (completed_jobs < n) {
        highest_priority_job = -1;
        highest_priority = INT_MAX;
        // Find the job with the highest priority
        for (i = 0; i < n; i++) {
            if (at[i] <= current_time && completed[i] == 0 && pt[i] < highest_priority) {
                highest_priority = pt[i];
                highest_priority_job = i;
            }
        }

        if (highest_priority_job == -1) {
            current_time++;
        } else {
            ft[highest_priority_job] = current_time + bt[highest_priority_job];
            tat[highest_priority_job] = ft[highest_priority_job] - at[highest_priority_job];
            wt[highest_priority_job] = tat[highest_priority_job] - bt[highest_priority_job]; // Corrected line

            completed[highest_priority_job] = 1;
            completed_jobs++;
            current_time = ft[highest_priority_job];
        }
    }
    // Calculate turnaround time and waiting time
    for (i = 0; i < n; i++) {
        totalTAT += tat[i];
        totalWT += wt[i];
    }
```

```
   // Print the output
   printf("\nJob\tArrival Time\tBurst Time\tPriority\tFinish Time\tTurnaround Time\tWaiting Time\n");
   for (i = 0; i < n; i++) {
       printf("Job%d\t%d\t\t%d\t\t%d\t\t%d\t\t%d\t\t%d\n", i + 1, at[i], bt[i], pt[i], ft[i], tat[i], wt[i]);
   }
   printf("\nAverage Turnaround Time: %.2f\n", totalTAT / n);
   printf("Average Waiting Time: %.2f\n", totalWT / n);


   return 0;
}
```

Sample Input and Output:



```
Enter number of jobs: 5
Arrival time of job[1]: 0
Burst time of job[1]: 5
Priority of job[1]: 2
Arrival time of job[2]: 1
Burst time of job[2]: 3
Priority of job[2]: 1
Arrival time of job[3]: 2
Burst time of job[3]: 8
Priority of job[3]: 4
Arrival time of job[4]: 3
Burst time of job[4]: 6
Priority of job[4]: 3
Arrival time of job[5]: 4
Burst time of job[5]: 4
Priority of job[5]: 2
```



```
Job     Arrival Time    Burst Time      Priority        Finish Time     Turnaround Time Waiting Time
Job1    0               5               2               5               5               0
Job2    1               3               1               8               7               4
Job3    2               8               4               26              24              16
Job4    3               6               3               18              15              9
Job5    4               4               2               12              8               4

Average Turnaround Time: 11.80
Average Waiting Time: 6.60
```

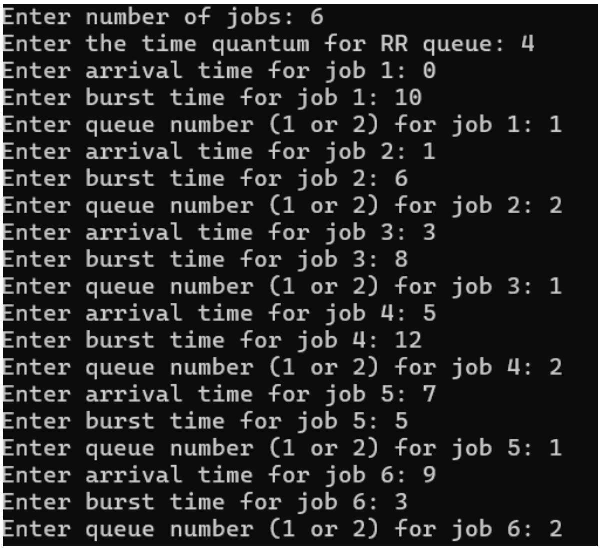
Experiment No.:6

Experiment Name: Write a C program to simulate the Multilevel Queue scheduling algorithms to find turnaround time and waiting time for a problem.

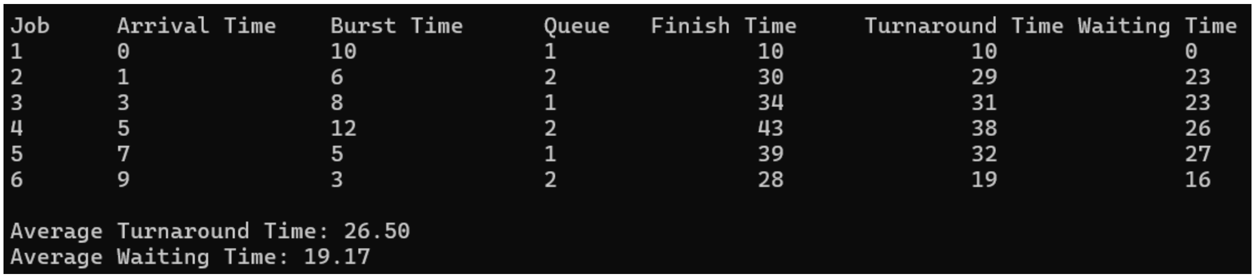
Source Code:

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_JOBS 10

typedef struct {
    int job_id;
    int arrival_time;
    int burst_time;
    int remaining_burst_time;
    int queue_num; // 1 or 2 in this example
    int finish_time;
    int turnaround_time;
    int waiting_time;
} Job;

void fcfs(Job jobs[], int n, int start_index, int end_index) {
    int current_time = jobs[start_index].arrival_time;
    for (int i = start_index; i <= end_index; i++) {
        if (current_time < jobs[i].arrival_time) {
            current_time = jobs[i].arrival_time;
        }
        jobs[i].finish_time = current_time + jobs[i].burst_time;
        jobs[i].turnaround_time = jobs[i].finish_time - jobs[i].arrival_time;
        jobs[i].waiting_time = jobs[i].turnaround_time - jobs[i].burst_time;
        current_time = jobs[i].finish_time;
    }
}

void round_robin(Job jobs[], int n, int start_index, int end_index, int time_quantum) {
    int current_time = 0;
    int completed_jobs = 0;
    int remaining_jobs = end_index - start_index + 1;
    int job_indices[MAX_JOBS];
    int job_count = 0;

    for (int i = start_index; i <= end_index; i++) {
        job_indices[job_count++] = i;
        if (jobs[i].arrival_time > current_time){
            current_time = jobs[i].arrival_time;
        }
    }
    while (completed_jobs < remaining_jobs) {
        for (int i = 0; i < job_count; i++) {
            int current_job_index = job_indices[i];
            if (jobs[current_job_index].remaining_burst_time > 0) {
                if (jobs[current_job_index].remaining_burst_time <= time_quantum) {
                    current_time += jobs[current_job_index].remaining_burst_time;
                    jobs[current_job_index].finish_time = current_time;
                    jobs[current_job_index].turnaround_time = jobs[current_job_index].finish_time -
jobs[current_job_index].arrival_time;
                    jobs[current_job_index].waiting_time = jobs[current_job_index].turnaround_time -
jobs[current_job_index].burst_time;
                    jobs[current_job_index].remaining_burst_time = 0;
                    completed_jobs++;
                } else {
                    current_time += time_quantum;
                    jobs[current_job_index].remaining_burst_time -= time_quantum;
                }
            }
        }
    }
}
int main() {
    Job jobs[MAX_JOBS];
    int n, time_quantum;
    printf("Enter number of jobs: ");
    scanf("%d", &n);
    printf("Enter the time quantum for RR queue: ");
    scanf("%d", &time_quantum);
    for (int i = 0; i < n; i++) {
        jobs[i].job_id = i + 1;
        printf("Enter arrival time for job %d: ", i + 1);
        scanf("%d", &jobs[i].arrival_time);
        printf("Enter burst time for job %d: ", i + 1);
        scanf("%d", &jobs[i].burst_time);
        jobs[i].remaining_burst_time = jobs[i].burst_time;
        printf("Enter queue number (1 or 2) for job %d: ", i + 1);
        scanf("%d", &jobs[i].queue_num);
    }

    int queue1_start = -1, queue1_end = -1, queue2_start = -1, queue2_end = -1;
    for (int i = 0; i < n; i++) {
        if (jobs[i].queue_num == 1) {
            if (queue1_start == -1) queue1_start = i;
            queue1_end = i;
        } else if (jobs[i].queue_num == 2) {
            if (queue2_start == -1) queue2_start = i;
            queue2_end = i;
        }
    }

    if (queue1_start != -1) {
        fcfs(jobs, n, queue1_start, queue1_end);
    }
    if (queue2_start != -1) {
        round_robin(jobs, n, queue2_start, queue2_end, time_quantum);
    }

    printf("\nJob\tArrival Time\tBurst Time\tQueue\tFinish Time\tTurnaround Time\tWaiting Time\n");
    for (int i = 0; i < n; i++) { printf("%d\t%d\t\t%d\t\t%d\t\t%d\t\t%d\t\t%d\n", jobs[i].job_id,
jobs[i].arrival_time, jobs[i].burst_time, jobs[i].queue_num, jobs[i].finish_time,
jobs[i].turnaround_time, jobs[i].waiting_time);

    }

    float total_tat = 0, total_wt = 0;

    for (int i = 0; i < n; i++) {

        total_tat += jobs[i].turnaround_time;

        total_wt += jobs[i].waiting_time;

    }

  printf("\nAverage Turnaround Time: %.2f\n", total_tat / n);

    printf("Average Waiting Time: %.2f\n", total_wt / n);

    return 0;

}
```

Sample Input and Output:



```
Enter number of jobs: 6
Enter the time quantum for RR queue: 4
Enter arrival time for job 1: 0
Enter burst time for job 1: 10
Enter queue number (1 or 2) for job 1: 1
Enter arrival time for job 2: 1
Enter burst time for job 2: 6
Enter queue number (1 or 2) for job 2: 2
Enter arrival time for job 3: 3
Enter burst time for job 3: 8
Enter queue number (1 or 2) for job 3: 1
Enter arrival time for job 4: 5
Enter burst time for job 4: 12
Enter queue number (1 or 2) for job 4: 2
Enter arrival time for job 5: 7
Enter burst time for job 5: 5
Enter queue number (1 or 2) for job 5: 1
Enter arrival time for job 6: 9
Enter burst time for job 6: 3
Enter queue number (1 or 2) for job 6: 2
```



```
Job     Arrival Time    Burst Time      Queue   Finish Time     Turnaround Time Waiting Time
1       0               10              1               10              10              0
2       1               6               2               30              29              23
3       3               8               1               34              31              23
4       5               12              2               43              38              26
5       7               5               1               39              32              27
6       9               3               2               28              19              16

Average Turnaround Time: 26.50
Average Waiting Time: 19.17
```