Name : Lakhan Kumawat
Roll No : 1906055
Branch : CSE-1
Course : CSL4404

Assignment

To Write Scheduling Algorithms in C for following :-

1. FCFS - First Come First Serve
2. SJF - Shortest Job First
3. RR - Round Robin

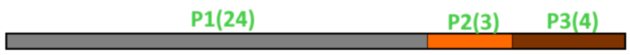
First Come First Serve

Given n processes with their burst times, the task is to find average waiting time and average turn around time using FCFS scheduling algorithm.
First in, first out (FIFO), also known as first come, first served (FCFS), is the simplest scheduling algorithm.
FIFO simply queues processes in the order that they arrive in the ready queue.

FCFS (Example)

Process	Duration	Oder	Arrival Time
P1	24	1	0
P2	3	2	0
P3	4	3	0

Gantt Chart :



P1 waiting time :	0
P2 waiting time :	24
P3 waiting time :	27

The Average waiting time :

(0+24+27)/3 = 17

In this, the process that comes first will be executed first
and next process starts only after the previous gets fully executed.

Name: Lakhan kumawat
Roll No: 1906055
Branch: CSE-1.

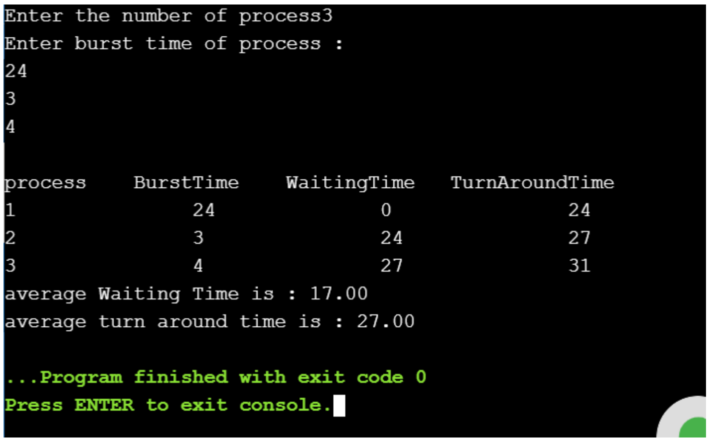
Course: CSL4404.

1. Write a program in C to implement First Come First Serve (FCFS) CPU Scheduling. Calculate average turnaround time and average waiting time.

```
#include <stdio.h>
int main(){
    printf("Enter the number of process");
    int nop;
    scanf("%d",&nop);
  int at[nop], bt[nop], wt[nop], tat[nop];
  float av, avgtat;
  printf(" Enter burst time of process: \n");
  for(int i=0; i<nop; i++){
      scanf("%d",&bt[i]);
          at[i]=i+1;
  }
  // waiting time
     wt[0] = 0;
     int temp=wt[0];
   for (int i=1; i< nop ; i++){
       wt[i] = temp+bt[i-1];
       temp += wt[i];
   }
     av= temp/nop;
     temp=0;
   for (int i=0; i<nop; i++){
     tat[i]= bt[i] + wt[i];
     temp += tat[i];
   }
```

Name: Lakhan Kumawat
Roll NO: 1906055
CSL4404.

```
avgtat = temp/nop;
 printf("\n");
     printf("process    BurstTime  WaitingTime  TurnAround
                                                 Time\n");
for (int i=0; i<nop; i++){
printf("%d\t\t%d \t\t %d \t\t %d \n", i+1, bt[i], wt[i],
                                         tat[i]);

}
printf ("average waiting time is: %0.2f \n",av);
prinf ("average Turn around time is: %0.2f", avgtat);
```



```
Enter the number of process3
Enter burst time of process :
24
3
4

process     BurstTime     WaitingTime    TurnAroundTime
1               24              0               24
2               3               24              27
3               4               27              31
average Waiting Time is : 17.00
average turn around time is : 27.00

...Program finished with exit code 0
Press ENTER to exit console.
```

Shortest Job First

Shortest job first(SJF) is a scheduling algorithm,

that is used to schedule processes in an operating system.

It is a very important topic in Scheduling when compared to

round-robin and FCFS Scheduling.

There are two types of SJF

Preemptive SJF

Non-Preemptive SJF

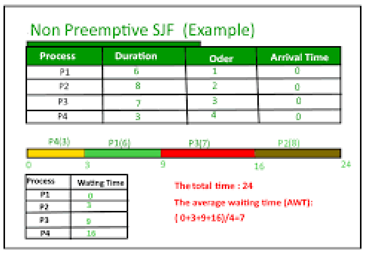
These algorithms schedule processes in the order in which the shortest job is done first.
It has a minimum average waiting time.

There are 3 factors to consider while solving SJF, they are

Burst Time
Average waiting time
Average turnaround time

Non-Preemptive Shortest Job First

Here is an example



Non Preemptive SJF (Example)

Process	Duration	Oder	Arrival Time
P1	6	1	0
P2	8	2	0
P3	7	3	0
P4	3	4	0

P4(3)
P1(6)
P3(7)
P2(8)
0
3
9
16
24

Process	Waiting Time
P1	0
P2	3
P3	9
P4	16

The total time : 24

The average waiting time (AWT):

(0+3+9+16)/4=7

Name: Lakhan kumawat
Roll No: 1906055
Branch: CSE-1

CSL4404.

2. write a program in C to Implement the Shortest Job First non-preemptive CPU scheduling. Calculate average turnaround time and average waiting time

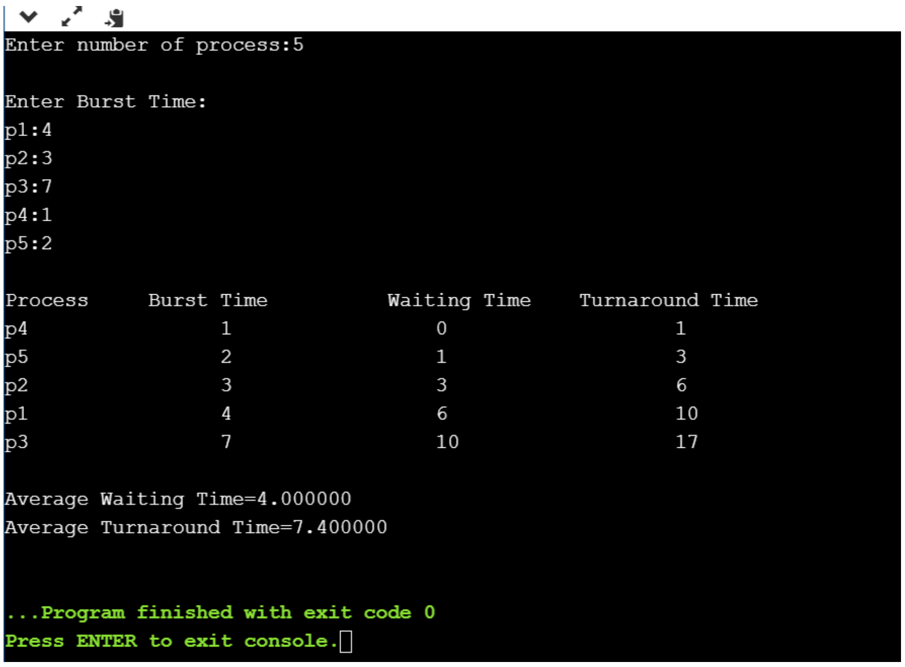
```
#include <stdio.h>
 int main()
{
   int bt[20],p[20], wt[20] ,tat[20], i,j,n,total=0 , pos, temp;

   float avg-wt , avg-tat ;
  printf (" Enter number of process : ");
   scanf ("%d",&n);

  printf ("Enter Burst Time: n");
   for (i=0; i<n;i++)
   {
      printf ("p %d:",i+1);
      scanf ("%d", & bt[i]);
       p[i] = i+1 ;
    }
   // sorting burst times.

    for (i=0; i<n;i++) {
       pos =i;
        for (j=i+1; j<n;j++){
              if (bt [j] < bt[pos] ) pos = j;

          temp = bt[i];
          bt[i] = bt[pos];
           bt[pos]= temp;

           temp =p[i];
           p[i] = p[pos];
```

Name: Lakhan Kumawat
Roll NO: 1906055
Branch: CSE-1

CSL4404

```
        P[pos]= temp;

}
    wt[0]=0;

  for (i=1; i<n; i++) {

    wt[i] = 0;
      for (j=0; j<i; j++) {
            wt[i] += bt[j];

            total += wt[i];

    }

    avg-wt =(float) total/n;
      total=0;

printf ("\n process \t Burst Time \t waiting Time  \t Turn around Time");

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

      tat [i]= bt[i] +wt[i];
        total+= tat[i];
        printf ("\np %d \t\t  %d\t\t %d \t\t\t", p[i], bt[i], wt[i],
          tat[i] ); }

  avg-tat =(float) total /n;

     printf (" \n\n Average waiting Time= %f", avg-wt);
    printf (" \n Average TurnAround Time= %f\n", avg-tat);

}
```

OUTPUT



```
Enter number of process:5

Enter Burst Time:
p1:4
p2:3
p3:7
p4:1
p5:2

Process     Burst Time          Waiting Time    Turnaround Time
p4                1                  0                  1
p5                2                  1                  3
p2                3                  3                  6
p1                4                  6                  10
p3                7                  10                 17

Average Waiting Time=4.000000
Average Turnaround Time=7.400000


...Program finished with exit code 0
Press ENTER to exit console.
```

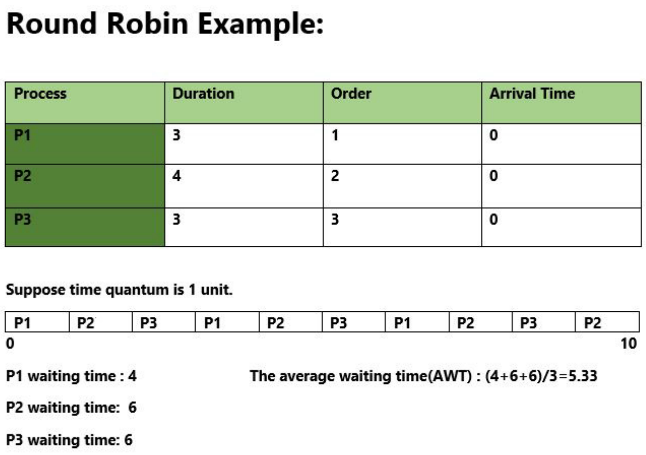
Round Robin

A round-robin is a CPU scheduling algorithm that shares equal portions of resources in circular orders to each process and handles all processes without prioritization.

In the round-robin, each process gets a fixed time interval of the slice to utilize the resources or execute its task called time quantum or time slice.

Some of the round-robin processes are pre-empted
if it executed in a given time slot, while the rest of the processes go back to
the ready queue and wait to run in a circular order with the scheduled time slot
until they complete their task.

It removes the starvation for each process to achieve
CPU scheduling by proper partitioning of the CPU.



Round Robin Example:

Process	Duration	Order	Arrival Time
P1	3	1	0
P2	4	2	0
P3	3	3	0

Suppose time quantum is 1 unit.

P1	P2	P3	P1	P2	P3	P1	P2	P3	P2

0 10

P1 waiting time : 4

P2 waiting time: 6

P3 waiting time: 6

The average waiting time(AWT) : (4+6+6)/3=5.33

Name: Lakhan Kumawat
Roll No: 1906055
CSL4404

3. Write a program in C to implement the Round Robin preemptive CPU Scheduling. Calculate average turnaround time and average waiting time. [Time quantum = 4]

```
# include <stdio.h>
# include <conio.h>

Void main ()
 {
    //initialize the variable name
    int i ,NOP, Sum=0, Count=0 ,y, quant, wt=0, tat=0, at[10], bt[10], bt[10],
     temp[10];
     float avg_wt, avg_tat;

     printf(" Total number of process in the System :");
    scanf(" %d ",&NOP);
      y= NOP; //Assign the number of process to variable y.
    //use loop to enter the details of the process
    for (i=0; i<NOP; i++)
     {
         printf (" \n Enter the Arrival and Burst time of process [%d ]\n",i+1);
         printf ("Arrival time is: \t ");
         scanf (" %d ",&at [i]);
         printf(" \n Burst time is: \t");
         scanf(" %d ", &bt[i]);
          temp[i] = bt[i] ;
       }
```

Name: Lakhan kumawat

Roll No: 1906055

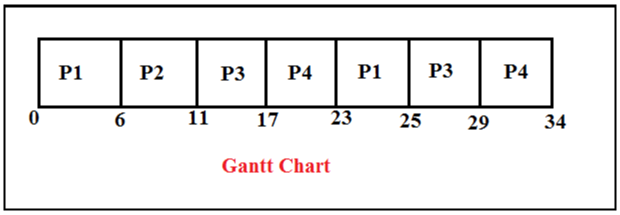
Branch: CSE-1

RR-②

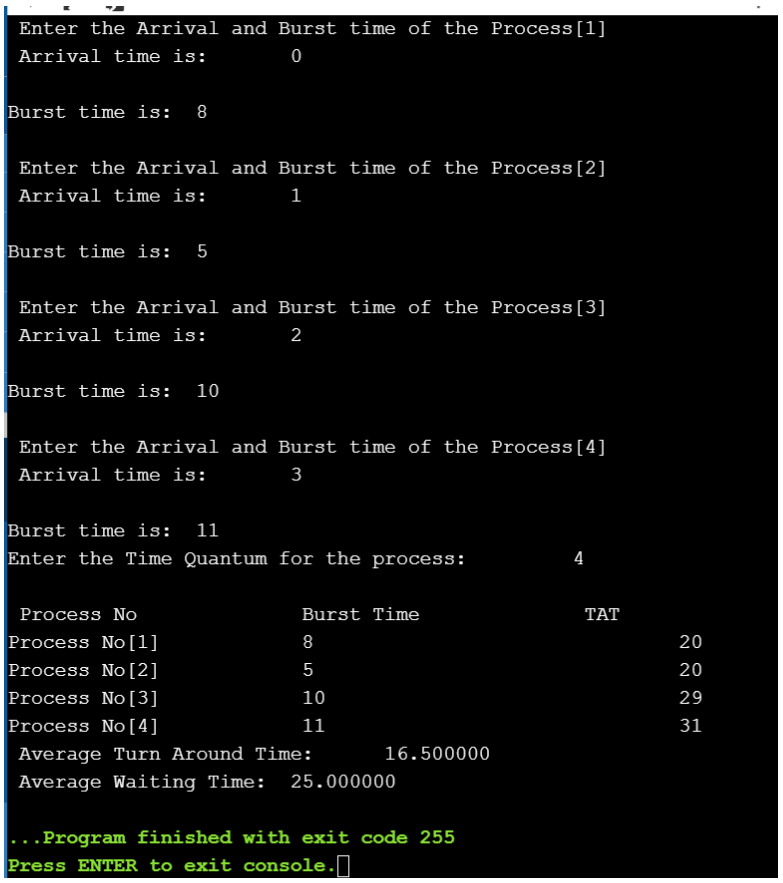
CSL4404.

```
printf ("Enter the time quantum for the process : \t");
scanf ("%d", &quant);

printf ("\n process No \t\t Burst Time \t\t TAT  \t\t Waiting Time");
for (sum=0; i=0; y!=0)
{ if(temp[i] <= quant && temp[i]>0)
  { sum += quant; }
else if ( temp[i]>0)
  {
    temp[i] -= quant;
     Sum  += quant;
  }
if (temp[i] ==0 && Count ==1) {
   y--;
  printf ("\n process No[%d] \t\t %d \t\t\t\t %d \t\t %d",
         i+1, bt[i], sum - at[i] , sum - at[i]-bt[i]);
  wt=wt+ sum - at[i] - bt[i];
  tat= tat +sum- at[i];
  Count=0 ;
}
if (i==NOP-1)  i= 0;
else if (at[i+1]<=sum)  i++;
else  i= 0;
avg_ wt= wt* 1.0/ NOP;  avg_tat =tat * 1.0/NOP;

printf ("\n Average Turn Around Time :\t%f", avg_wt );
printf ("\n Average Waiting Time : \t%f ", avg_ tat);
getch();
}
```

OUTPUTS



Gantt Chart



```
 Enter the Arrival and Burst time of the Process[1]
 Arrival time is:       0

Burst time is:  8

 Enter the Arrival and Burst time of the Process[2]
 Arrival time is:       1

Burst time is:  5

 Enter the Arrival and Burst time of the Process[3]
 Arrival time is:       2

Burst time is:  10

 Enter the Arrival and Burst time of the Process[4]
 Arrival time is:       3

Burst time is:  11
Enter the Time Quantum for the process:         4

 Process No              Burst Time              TAT
Process No[1]            8                               20
Process No[2]            5                               20
Process No[3]            10                              29
Process No[4]            11                              31
 Average Turn Around Time:      16.500000
 Average Waiting Time:  25.000000

...Program finished with exit code 255
Press ENTER to exit console.
```