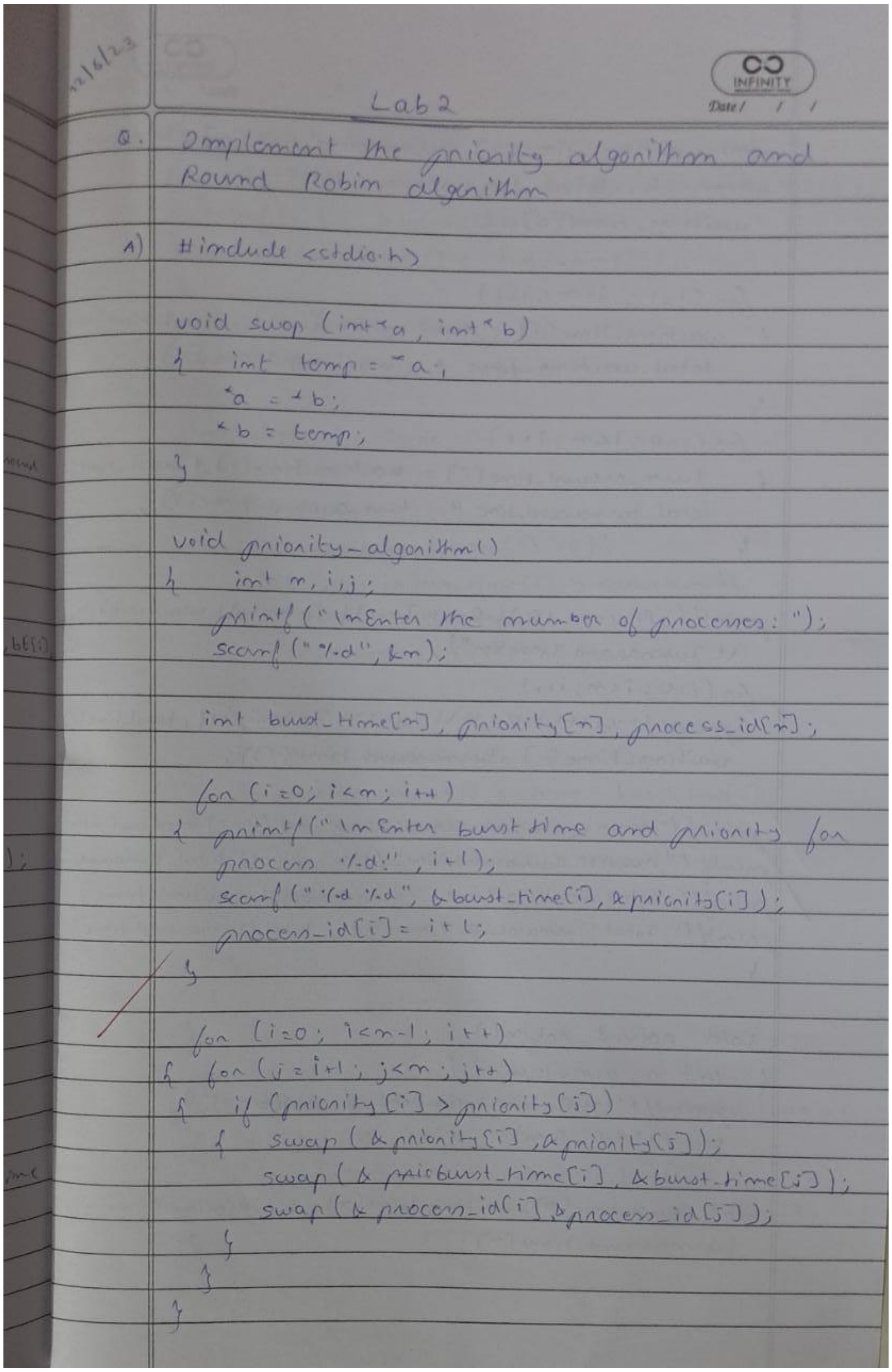
Priority and SJF



12/6/23

Lab 2

Date / / /

Q. Implement the priority algorithm and Round Robin algorithm

A)
```
#include <stdio.h>

void swap (int*a, int* b)
{   int temp = *a;
    *a = *b;
    *b = temp;
}

void priority_algorithm()
{   int n, i, j;
    printf("\nEnter the number of processes: ");
    scanf("%d", &n);

    int burst_time[n], priority[n], process_id[n];

    for (i=0; i<n; i++)
    { printf("\nEnter burst time and priority for
      process %d:", i+1);
      scanf("%d %d", &burst_time[i], &priority[i]);
      process_id[i] = i+1;
    }

    for (i=0; i<n-1; i++)
    { for (j=i+1; j<n; j++)
      {  if (priority[i] > priority[j])
         {  swap(&priority[i], &priority[j]);
            swap(&priobust_time[i], &burst_time[j]);
            swap(&process_id[i], &process_id[j]);
         }
      }
    }
```

Date / / /

```
int waiting_time[n], turnaround_time[n], total_wait
time =0, total_turnaround_time =0;
waiting_time[0] =0;

for (i=1; i<n; i++)
{ waiting_time[i] = waiting_time[i-1] + burst_time[i-1]
   total_waiting_time += waiting_time[i];
}
for (i=0; i<n; i++)
{  turn_around_time[i] = waiting_time[i] + burst_time[i]
   total_turnaround_time += turn_around_time[i];
}

printf("Process ID \t Burst Time\t Priority\t WaitingTime
 \t Turnaround Time\n");
for (i=0; i<n; i++)
  printf("%d\t\t %d \t\t %d\t\t %d\n", i+1, burst_time[i],
  waiting_time[i], turnaround_time[i]);

printf("Average Waiting Time : %f\n", ~~tot~~(float) total_waiting_time
printf(" Average Turnaround Time: %f\n", (float) total_turnaround
printf(" Total waiting Time: %d", ~~ta~~ total_waiting_time)
printf(" Total Turnaround Time: %d", total_turnaround_time)
}

void round_robin()
{ int n, quantum, i, j;
  printf("\n Enter number of process : ");
  scanf("%d", &n);

int burst_time[n], waiting_time[n], remaining_time[n],
turnaround_time[n];
```

```
for(i=0; i<n; i++)
{ printf("\n Enter burst time for process %d", i+1);
  scanf("%d", &burst_time[i]);
  remaining_time[i] = burst_time[i];
}
printf("\n Enter the quantum ");
scanf("%d", &quantum);

 int time =0, done =0;
 while (done != n)
{  for(i=0; i<n; i++)
  {   if ( remaining_time[i] >0){
        if (remaining_time[i] > quantum){
           time += quantum;
           remaining_time[i] -= quantum;
        }
       else
       {  time += remaining_time[i];
          waiting_time[i] = time - burst_time[i];
          remaining_time[i] =0;
          turnaround_time[i] = time;
          done ++;
       }
     }
   }
 }

float total_waiting_time =0 , total_turnaround_time = 0;
 for (i=0; i<n; i++){
   total_waiting_time += waiting_time[i];
   total_turnaround_time += turnaround_time[i];
 }
```

Date / / /

```
printf("PRocess ID\t BurstTime \t WaitingTime\t Turnaround tim
for (i=0; i<n; i++)
    printf("%d\t\t %d\t\t %d\t\t%d\n", i+1, burst_time[i],
    waiting-time[i], turnaround_time[i]);

printf("Average waiting Time : %f \n", total_waiting_time/n);
printf("Average turnaround time: %f\n", total_turnaround_time/n);
printf("Total waiting Time: %f \n", total-waiting-time);
printf("Total aturnaround time: %f \n", total_turnaround_time)
}

int main()
{ int choice;
  while (1)
  { printf("\n \n Enter 1 for Priority \n 2 - Round Robin\n3-Exi
    scanf("%d", &choice);
    switch (choice)
    { case 1 : priority_alogarithm(); break;
      case 2 : round_robin(); break;
      Case 3 : exit(0);
      default: Iprintf("\n Invalid Input");
    }
  } return 0; }
```

Output:

Priority

```
Enter number of processes : 3
Enter burst time and priority for process 1 :  4 1
Enter burst time and priority for process 2 :  3 2
Enter burst time and priority for process 3 :  5 3
Process ID  Burst Time  Priority  waiting Time  Turnaround Time
   1           4           1           0             4
   2           3           2           4             7
   3           5           3           7             12
```

23/6/23

Date / /

Average Waiting Time : 3.66
Average Turnaround Time : 7.66
Total Waiting Time : 11
Total Turnaround Time : 23

Round Rot Tracing
process , priority ((3, 3) , (2, 2) , (1, 1))
Sorting : (1, 1) , (2, 2) , (3, 3)
Hence above output

Round Robin :
Enter number of process : 3
Enter burst time for process 1 : 4
Enter burst time for process 2 : 3
Enter burst time for process 3 : 5

Process ID	Burst Time	Waiting Time	Turnaround Time
1	4	4	8
2	3	6	9
3	5	7	12

Average Waiting Time : 5.66
Average Turnaround Time : 9.66
Total Waiting Time : 17.00
Total Turnaround Time : 29.00

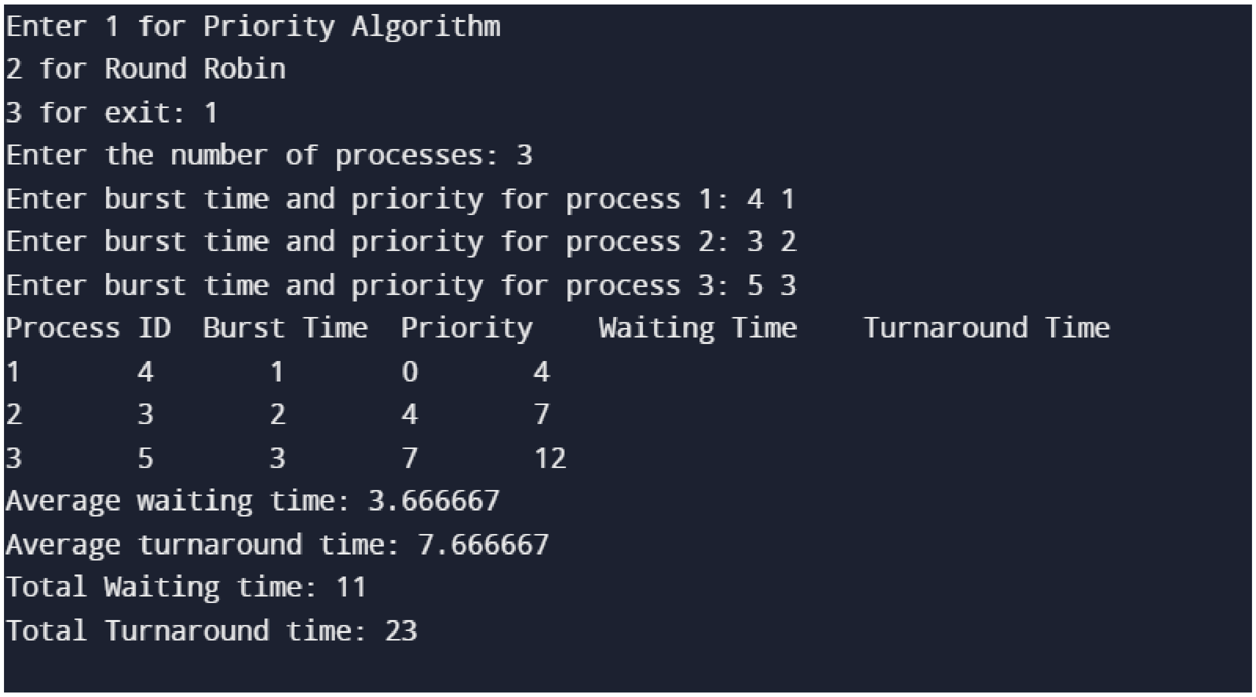
Tracing

Burst	4	3	5	2	1	3	0	0	0
	P1	P2	P3	P1	P2	P3	P1	P2	P3
Rem	4-2=2	3-2=1	5-2=3	2-2=0	1-1=0	3-2=1	0	0	1-1=0
wait	2	2	2	2	1	2	0	0	1

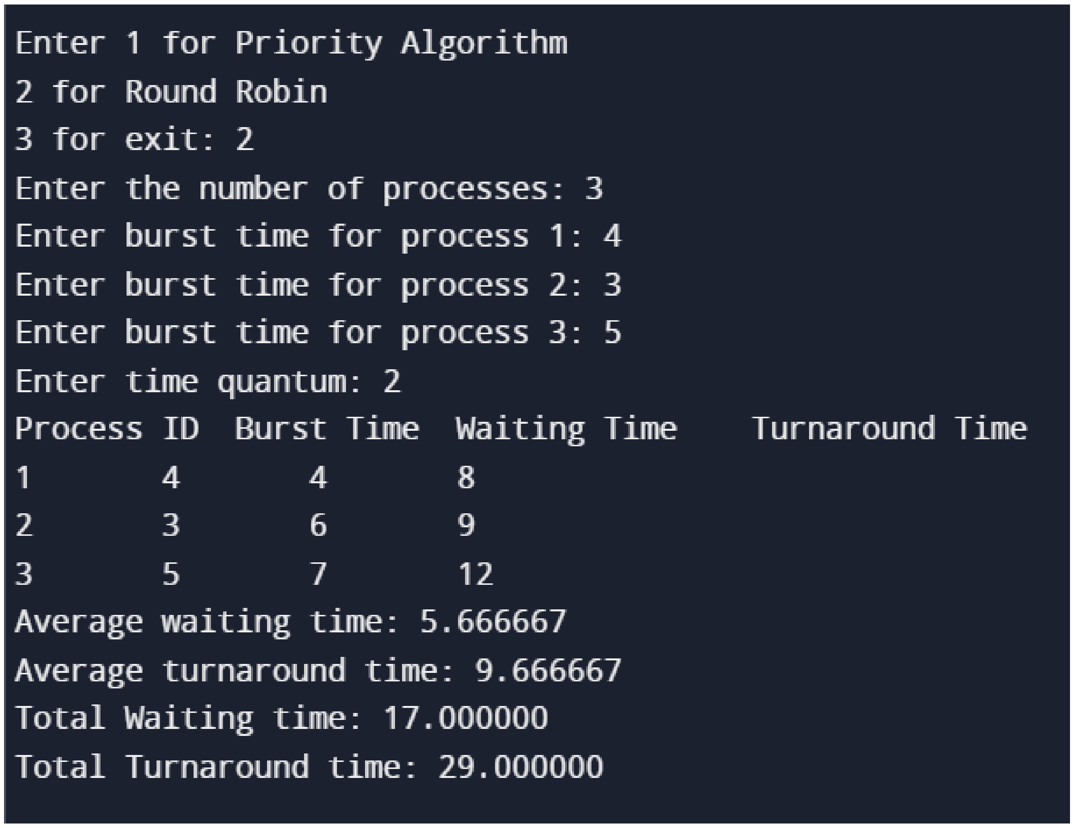
P1 (Time) = ~~6~~ 8
P2 (Time) = 9
P3 (Time) = 12

N 23/6/23

wait Time = Total - Burst
P1 = 8 - 4 = 4
P2 = 9 - 3 = 6
P3 = 12 - 5 = 7



```
Enter 1 for Priority Algorithm
2 for Round Robin
3 for exit: 1
Enter the number of processes: 3
Enter burst time and priority for process 1: 4 1
Enter burst time and priority for process 2: 3 2
Enter burst time and priority for process 3: 5 3
Process ID  Burst Time  Priority    Waiting Time    Turnaround Time
1       4       1       0       4
2       3       2       4       7
3       5       3       7       12
Average waiting time: 3.666667
Average turnaround time: 7.666667
Total Waiting time: 11
Total Turnaround time: 23
```



```
Enter 1 for Priority Algorithm
2 for Round Robin
3 for exit: 2
Enter the number of processes: 3
Enter burst time for process 1: 4
Enter burst time for process 2: 3
Enter burst time for process 3: 5
Enter time quantum: 2
Process ID  Burst Time  Waiting Time    Turnaround Time
1       4       4       8
2       3       6       9
3       5       7       12
Average waiting time: 5.666667
Average turnaround time: 9.666667
Total Waiting time: 17.000000
Total Turnaround time: 29.000000
```