DHRUV GARG 16 BCE 1190

SCHEDULING ALGORITHMS

LAB 6

In Student Xerox, to take copy of the drafts, token system is being followed. Assume that there are 5 customers waiting in a queue with token numbers.

Find algorithms (i) favours the order (ii) favours the customer and compare the waiting time involved in both the algorithms.

Customer. No	Time required to take copy	Tokens
T1	10	3
T2	8	2
T3	2	1
T4	6	5
T5	5	4

CODE (*Has been programmed entirely by me)

```
#include<stdio.h>


int fcfsScheduling(int *process, int *burst, int *priority, int num)
{
   int i, j, waiting[20], turnaround[20], totWait = 0, totTurnaround = 0, counter = 0;
   int tempProcess[20], tempBurst[20], tempPriority[20], curr = 0, totTime = 0;
   float avgWait = 0, avgTurnaround = 0;
   float n = num * 1.0;

   for(i = 0; i < n; i++)
   {
      tempProcess[i] = process[i];
      tempBurst[i] = burst[i];
      tempPriority[i] = priority[i];
   }
   printf("\n\n\tTHE PROCESSES ARE SCHEDULED AS FOLLOWS : \n\n");
   printf("\t------------------------------------------------------------");
   printf("\n\tPROCESS\t\tPRIORITY\tREMAINING TIME\tTIME ELAPSED\n");
   printf("\t------------------------------------------------------------\n");
   for(i = 0; counter != num; i++)
   {
      for(j = 0; j < n; j++)
      {
         if(tempBurst[j] > 0)
         {
            curr = j;
            break;
         }
      }
      tempBurst[curr] -= 1;
      printf("\tP[%d]\t\t%d\t\t%d\t\tTime : %d\n", tempProcess[curr], tempPriority[curr], tempBurst[curr], i+1);
      totTime = i+1;
      if(tempBurst[curr] == 0)
      {
         counter++;
         waiting[curr] = totTime - burst[curr];
         turnaround[curr] = totTime;
         printf("\n\tCOUNTER : %d\n\n", counter);
      }
   }
   printf("\n\t\tALL PROCESSES HAVE COMPLETED EXECUTION\n\n\n");
   printf("\t------------------------------------------------------------\n");
   printf("\tProcess\t\tPriority\tWaiting\t\tTurnaround\n");
   printf("\t------------------------------------------------------------\n");
   for(i = 0; i < n; i++)
   {
      totWait += waiting[i];
      totTurnaround += turnaround[i];
      printf("\tP[%d]\t\t%d\t\t%d\t\t%d\n", tempProcess[i], tempPriority[i], waiting[i], turnaround[i]);
```

```
    }
    avgWait = totWait/n;
    avgTurnaround = totTurnaround/n;
    printf("\n\n\tThe average waiting time : %f\n", avgWait);
    printf("\tThe average turnaround time : %f\n",
avgTurnaround);
    return 0;
}

int sjfScheduling(int *process, int *burst, int *priority, int num)
{
    int i, j, temp = 0, index = 0, min = 0;
    int ascProcess[20], corresBurst[20];
    int tempProcess[20], tempBurst[20], tempPriority[20];
    for(i = 0; i < num; i++)
    {
        tempProcess[i] = process[i];
        tempBurst[i] = burst[i];
        tempPriority[i] = priority[i];
    }
    for(i = 0; i < num; i++)
    {
        temp = tempBurst[i];
        index = i;
        min = tempBurst[i];
        for(j = i+1; j < num; j++)
        {
            if((tempBurst[j] < temp)&&(tempBurst[j] < min))
            {
                index = j;
                min = tempBurst[j];
            }
        }
        corresBurst[i] = tempBurst[index];
        tempBurst[i] = tempBurst[index];
        tempBurst[index] = temp;
        temp = tempPriority[i];
        tempPriority[i] = tempPriority[index];
        tempPriority[index] = temp;
        temp = tempProcess[i];
        tempProcess[i] = tempProcess[index];
        tempProcess[index] = temp;
        ascProcess[i] = tempProcess[i];
    }

    printf("\n\tTHE ESTABLISHED SEQUENCE OF EXECUTION\n");
    for(i = 0; i < num; i++)
    {
        printf("\n\tProcess number : %d\n", ascProcess[i]);
        printf("\tCorresponding burst : %d\n", corresBurst[i]);
    }
    fcfsScheduling(tempProcess, tempBurst, tempPriority, num);
    return 0;
}

int priorityScheduling(int *process, int *burst, int *priority, int
num)
{
    int i, j, waiting = 0, turnaround = 0, totWait = 0,
totTurnaround = 0;
    float avgWait = 0, avgTurnaround = 0;
    float n = num * 1.0;
    int tempProcess[20], tempBurst[20], tempPriority[20];
    printf("\n\tTHE ESTABLISHED SEQUENCE OF EXECUTION\n");
    for(i = 0; i < n; i++)
    {
        for(j = 0; j < n; j++)
        {
            if(i+1 == priority[j])
            {
                tempProcess[i] = process[j];
                tempBurst[i] = burst[j];
```

```
            tempPriority[i] = i+1;
            printf("\n\tPRIORITY : %d", tempPriority[i]);
            printf("\n\tProcess number : %d\n", tempProcess[i]);
            printf("\tCorresponding burst : %d\n\n",
tempBurst[i]);
         }
         else
            continue;
      }
   }
   fcfsScheduling(tempProcess, tempBurst, tempPriority, num);
   return 0;
}

int rrScheduling(int *process, int *burst, int *priority, int num,
int q)
{
   int i, totWait = 0, totTurnaround = 0, totTime = 0, diff = 0,
counter = 0;
   int waiting[20], turnaround[20], tempProcess[20],
tempBurst[20], flag = 0;
   float n, avgWait = 0, avgTurnaround = 0;
   n = num * 1.0;

   for(i = 0; i < n; i++)
   {
      tempProcess[i] = process[i];
      tempBurst[i] = burst[i];
   }
   printf("\n\tTHE ESTABLISHED SEQUENCE OF
EXECUTION\n\n");
   printf("\t-------------------------------------------");
   printf("\n\tPROCESS\t\tREMAINING TIME\tTIME
ELAPSED\n");
   printf("\t-------------------------------------------\n");
   while(flag == 0)
   {
      for(i = 0; i < n; i++)
      {
         if((tempBurst[i]-q) > 0)
         {
            tempBurst[i] -= q;
            totTime += q;
            printf("\tP[%d]\t\t%d\t\t", tempProcess[i],
tempBurst[i]);
            printf("Time : %d\n", totTime);
         }
         else if((tempBurst[i] - q) == 0)
         {
            tempBurst[i] -= q;
            totTime += q;
            waiting[i] = totTime - burst[i];
            turnaround[i] = totTime;
            counter++;
            printf("\tP[%d]\t\t%d\t\t", tempProcess[i],
tempBurst[i]);
            printf("Time : %d\n", totTime);
            printf("\n\tCOUNTER : %d\n\n", counter);
         }
         else if(((tempBurst[i]-q) < 0)&&(tempBurst[i] > 0))
         {
            diff = tempBurst[i];
            tempBurst[i] = 0;
            totTime += diff;
            waiting[i] = totTime - burst[i];
            turnaround[i] = totTime;
            counter++;
            printf("\tP[%d]\t\t%d\t\t", tempProcess[i],
tempBurst[i]);
            printf("Time : %d\n", totTime);
            printf("\n\tCOUNTER : %d\n\n", counter);
         }
         if(counter == num)
         {
```

```
            printf("\n\t  ALL PROCESSES HAVE COMPLETED EXECUTION\n\n");
            flag = 1;
            break;
        }
    }
  }
  printf("\t----------------------------------------\n");
  printf("\tProcess\t\tWaiting\t\tTurnaround\n");
  printf("\t----------------------------------------\n");
  for(i = 0; i < num; i++)
  {
    totWait += waiting[i];
    totTurnaround += turnaround[i];
    printf("\tP[%d]\t\t%d\t\t%d\n", i+1, waiting[i], turnaround[i]);
  }
  avgWait = totWait/n;
  avgTurnaround = totTurnaround/n;
  printf("\n\n\tThe average waiting time in RR : %f", avgWait);
  printf("\n\tThe average turnaround time in RR : %f\n", avgTurnaround);
  return 0;
}


int sjrScheduling(int *process, int *burst, int *priority, int num)
{
  int i, j, totWait = 0, totTurnaround = 0, totTime = 0, counter = 0, smallest = 0;
  float n, avgWait = 0, avgTurnaround = 0;
  int flag = 0, tempProcess[20], tempBurst[20], arrival[20], index = 0;
  n = num * 1.0;
  int turnaround[20], waiting[20];
  printf("\n\n\t****************** \n");
  printf("\n\tNOTE : FOR SJR scheduling, the PRIORITY in the \n\t");
  printf("      given question is taken as PROCESS ARRIVAL TIME.\n");
  printf("\n\t****************** \n\n");
  printf("\n\tTHE ESTABLISHED SEQUENCE OF EXECUTION\n\n");
  printf("\t----------------------------------------");
  printf("\n\tPROCESS\t\tREMAINING TIME\tTIME ELAPSED\n");
  printf("\t----------------------------------------\n");
  for(i = 0; i < n; i++)
  {
    tempProcess[i] = process[i];
    tempBurst[i] = burst[i];
    arrival[i] = priority[i];
  }
  tempBurst[19] = 999;
  for(i = 0; counter != n; i++)
  {
    smallest = 19;
    for(j = 0; j < n; j++)
    {
      if((arrival[j] <= i) && (tempBurst[j] < tempBurst[smallest]) && (tempBurst[j] > 0))
      {
        smallest = j;
      }
    }
    tempBurst[smallest] -= 1;
    if(smallest == 19)
    {
      printf("\tIDLE\t\tNA\t\tTime : %d\n", i+1);
    }
    else
      printf("\tP[%d]\t\t%d\t\tTime : %d\n", smallest+1, tempBurst[smallest], i+1);
    if(tempBurst[smallest] == 0)
    {
      counter++;
```

```
            totTime = i + 1;
            waiting[smallest] = totTime - arrival[smallest] - burst[smallest];
            turnaround[smallest] = totTime - arrival[smallest];
            printf("\n\tCOUNTER : %d\n\n", counter);
        }
    }
    printf("\n\t  ALL PROCESSES HAVE COMPLETED EXECUTION\n\n");
    printf("\t----------------------------------------\n");
    printf("\tProcess\t\tWaiting\t\tTurnaround\n");
    printf("\t----------------------------------------\n");
    for(i = 0; i < num; i++)
    {
        totWait += waiting[i];
        totTurnaround += turnaround[i];
        printf("\tP[%d]\t\t%d\t\t%d\n", i+1, waiting[i], turnaround[i]);
    }
    avgWait = totWait/n;
    avgTurnaround = totTurnaround/n;
    printf("\n\n\tThe average waiting time : %f", avgWait);
    printf("\n\tThe average turnaround time : %f\n", avgTurnaround);
    return 0;
}

int dispProcess(int *process, int *burst, int *priority, int num)
{
    int i;
    printf("\n\n\tTHE PROCESSES ENTERED ARE AS FOLLOWS : \n");
    printf("\t-------------------------------------\n");
    printf("\tProcess\t\tBurst\tPriority\n");
    printf("\t-------------------------------------\n");
    for(i = 0; i < num; i++)
    {
        printf("\tP[%d]\t\t%d\t%d\n", process[i], burst[i], priority[i]);
    }
    printf("\t-------------------------------------\n");

    return 0;
}

int main()
{
    int process[20], burst[20], priority[20], num = 0, q = 0;
    printf("\n\tEnter the number of processes : ");
    scanf("%d", &num);
    int i = 0;
    printf("\n\tEnter the process number, burst time and priority : \n");
    for(i = 0; i < num; i++)
    {
        printf("\t");
        scanf("%d%d%d", &process[i], &burst[i], &priority[i]);
    }

    dispProcess(process, burst, priority, num);
    int ch = 10;
    do
    {
        printf("\n\n\t-------------------------------------\n");
        printf("\t   CHOOSE A SCHEDULING ALGORITHM \n");
        printf("\t-------------------------------------\n");
        printf("\t1. FCFS - non pre-emptive\n");
        printf("\t2. SJF - non pre-emptive\n");
        printf("\t3. Priority - pre-emptive\n");
        printf("\t4. SJR - pre-emptive\n");
        printf("\t5. Round Robin - pre-emptive\n");
        printf("\t6. Exit\n");
        printf("\n\tEnter your choice : ");
        scanf("%d", &ch);
```

```
    printf("\n");
    switch(ch)
    {
    case 1 :
    {
       dispProcess(process, burst, priority, num);
       printf("\n\t--------- FIRST COME FIRST SERVE SCHEDULING ----------\n");
       fcfsScheduling(process, burst, priority, num);
       printf("\n\n\t--------- FIRST COME FIRST SERVE SCHEDULING ----------\n\n\n");
       break;
    }
    case 2 :
    {
       dispProcess(process, burst, priority, num);
       printf("\n\t---------- SHORTEST JOB FIRST SCHEDULING ----------\n");
       sjfScheduling(process, burst, priority, num);
       printf("\n\t---------- SHORTEST JOB FIRST SCHEDULING ----------\n\n\n");
       break;
    }
    case 3 :
    {
       dispProcess(process, burst, priority, num);
       printf("\n\t---------- PRIORITY SCHEDULING ----------\n");
       priorityScheduling(process, burst, priority, num);
       printf("\n\t---------- PRIORITY SCHEDULING ----------\n\n\n");
       break;
    }
    case 4 :
    {
       dispProcess(process, burst, priority, num);
       printf("\n\t---------- SHORTEST JOB REMAINING SCHEDULING ---------\n");
       sjrScheduling(process, burst, priority, num);
       printf("\n\t---------- SHORTEST JOB REMAINING SCHEDULING ---------\n\n\n");
       break;
    }
    case 5 :
    {
       dispProcess(process, burst, priority, num);
       printf("\n\t--------------- ROUND ROBIN SCHEDULING ----------\n");
       printf("\n\n\tENTER QUANTUM VALUE : ");
       scanf("%d", &q);
       rrScheduling(process, burst, priority, num, q);
       printf("\n\t--------------- ROUND ROBIN SCHEDULING ----------\n\n\n");
       break;
    }
    case 6 :
    {
       printf("\n\n\t-----------------------------------\n");
       printf("\t\t  END OF PROGRAM\n");
       printf("\t-----------------------------------\n\n");
       break;
    }
    }
  }
  while(ch != 6);
}
```

SCREENSHOTS

Figure 1: Take the process number, burst time and process priority as input from the user

```
dhruv@dhruv-Inspiron-5559:~$ cd Documents
dhruv@dhruv-Inspiron-5559:~/Documents$ gcc -o scheduling scheduling.c
dhruv@dhruv-Inspiron-5559:~/Documents$ ./scheduling

        Enter the number of processes : 5

        Enter the process number, burst time and priority :
        1       10      3
        2       8       2
        3       2       1
        4       6       5
        5       5       4


        THE PROCESSES ENTERED ARE AS FOLLOWS :
        ----------------------------------
        Process         Burst   Priority
        ----------------------------------
        P[1]            10      3
        P[2]            8       2
        P[3]            2       1
        P[4]            6       5
        P[5]            5       4
        ----------------------------------


        ----------------------------------
           CHOOSE A SCHEDULING ALGORITHM
        ----------------------------------
        1. FCFS - non pre-emptive
        2. SJF - non pre-emptive
        3. Priority - pre-emptive
        4. SJR - pre-emptive
        5. Round Robin - pre-emptive
        6. Exit

        Enter your choice : 1
```

Figure 2: (On selecting option 1 form menu) FCFS scheduling Gantt chart and execution time

```
        --------- FIRST COME FIRST SERVE SCHEDULING ----------


        THE PROCESSES ARE SCHEDULED AS FOLLOWS :

        ------------------------------------------------------------
        PROCESS         PRIORITY        REMAINING TIME  TIME ELAPSED
        ------------------------------------------------------------
        P[1]            3               9               Time : 1
        P[1]            3               8               Time : 2
        P[1]            3               7               Time : 3
        P[1]            3               6               Time : 4
        P[1]            3               5               Time : 5
        P[1]            3               4               Time : 6
        P[1]            3               3               Time : 7
        P[1]            3               2               Time : 8
        P[1]            3               1               Time : 9
        P[1]            3               0               Time : 10

        COUNTER : 1

        P[2]            2               7               Time : 11
        P[2]            2               6               Time : 12
        P[2]            2               5               Time : 13
        P[2]            2               4               Time : 14
        P[2]            2               3               Time : 15
        P[2]            2               2               Time : 16
        P[2]            2               1               Time : 17
        P[2]            2               0               Time : 18

        COUNTER : 2

        P[3]            1               1               Time : 19
        P[3]            1               0               Time : 20

        COUNTER : 3
```

```
P[4]            5               5               Time : 21
P[4]            5               4               Time : 22
P[4]            5               3               Time : 23
P[4]            5               2               Time : 24
P[4]            5               1               Time : 25
P[4]            5               0               Time : 26

COUNTER : 4

P[5]            4               4               Time : 27
P[5]            4               3               Time : 28
P[5]            4               2               Time : 29
P[5]            4               1               Time : 30
P[5]            4               0               Time : 31

COUNTER : 5

        ALL PROCESSES HAVE COMPLETED EXECUTION


------------------------------------------------------------
Process         Priority        Waiting         Turnaround
------------------------------------------------------------
P[1]            3               0               10
P[2]            2               10              18
P[3]            1               18              20
P[4]            5               20              26
P[5]            4               26              31

The average waiting time : 14.800000
The average turnaround time : 21.000000


--------- FIRST COME FIRST SERVE SCHEDULING ----------
```

ANALYSIS of first come first serve scheduling

The first person or process to arrive (First In) is the first one to be dealt with (First Out).

This scheduling method is non pre-emptive, that is, the process will run until it finishes.

Because of this non pre-emptive scheduling, short processes which are at the back of the queue have to wait for the long process at the front to finish.

Figure 3: Algorithm menu displayed after execution of FCFS algorithm.

```
-----------------------------------
   CHOOSE A SCHEDULING ALGORITHM
-----------------------------------
1. FCFS - non pre-emptive
2. SJF - non pre-emptive
3. Priority - pre-emptive
4. SJR - pre-emptive
5. Round Robin - pre-emptive
6. Exit

Enter your choice : 2
```

Figure 4: (On selecting option 2 form menu) SJF scheduling Gantt chart and execution time

```
------------ SHORTEST JOB FIRST SCHEDULING -----------

THE ESTABLISHED SEQUENCE OF EXECUTION

Process number : 3
Corresponding burst : 2

Process number : 5
Corresponding burst : 5

Process number : 4
Corresponding burst : 6

Process number : 2
Corresponding burst : 8

Process number : 1
Corresponding burst : 10


THE PROCESSES ARE SCHEDULED AS FOLLOWS :

------------------------------------------------------------
PROCESS         PRIORITY        REMAINING TIME  TIME ELAPSED
------------------------------------------------------------
P[3]            1               1               Time : 1
P[3]            1               0               Time : 2

COUNTER : 1

P[5]            4               4               Time : 3
P[5]            4               3               Time : 4
P[5]            4               2               Time : 5
P[5]            4               1               Time : 6
P[5]            4               0               Time : 7

COUNTER : 2
```

```
P[4]            5               5               Time : 8
P[4]            5               4               Time : 9
P[4]            5               3               Time : 10
P[4]            5               2               Time : 11
P[4]            5               1               Time : 12
P[4]            5               0               Time : 13

COUNTER : 3

P[2]            2               7               Time : 14
P[2]            2               6               Time : 15
P[2]            2               5               Time : 16
P[2]            2               4               Time : 17
P[2]            2               3               Time : 18
P[2]            2               2               Time : 19
P[2]            2               1               Time : 20
P[2]            2               0               Time : 21

COUNTER : 4

P[1]            3               9               Time : 22
P[1]            3               8               Time : 23
P[1]            3               7               Time : 24
P[1]            3               6               Time : 25
P[1]            3               5               Time : 26
P[1]            3               4               Time : 27
P[1]            3               3               Time : 28
P[1]            3               2               Time : 29
P[1]            3               1               Time : 30
P[1]            3               0               Time : 31

COUNTER : 5


        ALL PROCESSES HAVE COMPLETED EXECUTION
```

```
-------------------------------------------------------------
Process         Priority        Waiting         Turnaround
-------------------------------------------------------------
P[3]            1               0               2
P[5]            4               2               7
P[4]            5               7               13
P[2]            2               13              21
P[1]            3               21              31


The average waiting time : 8.600000
The average turnaround time : 14.800000

------------ SHORTEST JOB FIRST SCHEDULING -----------
```

ANALYSIS of shortest job first scheduling

SJF is a scheduling policy that selects the waiting process with the smallest execution time to execute next.

However, it may cause starvation if shorter processes keep coming. This problem can be solved using the concept of aging.

Figure 5: (On selecting option 3 form menu) PRIORITY scheduling Gantt chart and execution time

```
----------------- PRIORITY SCHEDULING ----------------

THE ESTABLISHED SEQUENCE OF EXECUTION

PRIORITY : 1
Process number : 3
Corresponding burst : 2


PRIORITY : 2
Process number : 2
Corresponding burst : 8


PRIORITY : 3
Process number : 1
Corresponding burst : 10


PRIORITY : 4
Process number : 5
Corresponding burst : 5


PRIORITY : 5
Process number : 4
Corresponding burst : 6



THE PROCESSES ARE SCHEDULED AS FOLLOWS :

-------------------------------------------------------------
PROCESS         PRIORITY        REMAINING TIME  TIME ELAPSED
-------------------------------------------------------------
P[3]            1               1               Time : 1
P[3]            1               0               Time : 2

COUNTER : 1
```

```
P[2]            2               7               Time : 3
P[2]            2               6               Time : 4
P[2]            2               5               Time : 5
P[2]            2               4               Time : 6
P[2]            2               3               Time : 7
P[2]            2               2               Time : 8
P[2]            2               1               Time : 9
P[2]            2               0               Time : 10

COUNTER : 2

P[1]            3               9               Time : 11
P[1]            3               8               Time : 12
P[1]            3               7               Time : 13
P[1]            3               6               Time : 14
P[1]            3               5               Time : 15
P[1]            3               4               Time : 16
P[1]            3               3               Time : 17
P[1]            3               2               Time : 18
P[1]            3               1               Time : 19
P[1]            3               0               Time : 20

COUNTER : 3

P[5]            4               4               Time : 21
P[5]            4               3               Time : 22
P[5]            4               2               Time : 23
P[5]            4               1               Time : 24
P[5]            4               0               Time : 25

COUNTER : 4

P[4]            5               5               Time : 26
P[4]            5               4               Time : 27
P[4]            5               3               Time : 28
P[4]            5               2               Time : 29
P[4]            5               1               Time : 30
P[4]            5               0               Time : 31

COUNTER : 5
```

```
        ALL PROCESSES HAVE COMPLETED EXECUTION


------------------------------------------------------------
Process         Priority        Waiting         Turnaround
------------------------------------------------------------
P[3]            1               0               2
P[2]            2               2               10
P[1]            3               10              20
P[5]            4               20              25
P[4]            5               25              31


The average waiting time : 11.400000
The average turnaround time : 17.600000

----------------- PRIORITY SCHEDULING ----------------
```

ANALYSIS of priority scheduling

Priority scheduling algorithm is one of the most common scheduling algorithms in batch computing systems.

Each process is assigned a priority. Process with the highest priority is to be executed first and so on.

Processes with the same priority are executed on first come first served basis.

Figure 6: (On selecting option 4 form menu) SJR scheduling Gantt chart and execution time

```
---------- SHORTEST JOB REMAINING SCHEDULING ---------


******************

NOTE : FOR SJR scheduling, the PRIORITY in the
       given question is taken as PROCESS ARRIVAL TIME.

******************


THE ESTABLISHED SEQUENCE OF EXECUTION

-----------------------------------------
PROCESS         REMAINING TIME  TIME ELAPSED
-----------------------------------------
IDLE            NA              Time : 1
P[3]            1               Time : 2
P[3]            0               Time : 3

COUNTER : 1

P[2]            7               Time : 4
P[5]            4               Time : 5
P[5]            3               Time : 6
P[5]            2               Time : 7
P[5]            1               Time : 8
P[5]            0               Time : 9

COUNTER : 2

P[4]            5               Time : 10
P[4]            4               Time : 11
P[4]            3               Time : 12
P[4]            2               Time : 13
P[4]            1               Time : 14
P[4]            0               Time : 15

COUNTER : 3
```

```
P[2]            6               Time : 16
P[2]            5               Time : 17
P[2]            4               Time : 18
P[2]            3               Time : 19
P[2]            2               Time : 20
P[2]            1               Time : 21
P[2]            0               Time : 22

COUNTER : 4

P[1]            9               Time : 23
P[1]            8               Time : 24
P[1]            7               Time : 25
P[1]            6               Time : 26
P[1]            5               Time : 27
P[1]            4               Time : 28
P[1]            3               Time : 29
P[1]            2               Time : 30
P[1]            1               Time : 31
P[1]            0               Time : 32

COUNTER : 5


  ALL PROCESSES HAVE COMPLETED EXECUTION

-----------------------------------------
Process         Waiting         Turnaround
-----------------------------------------
P[1]            19              29
P[2]            12              20
P[3]            0               2
P[4]            4               10
P[5]            0               5


The average waiting time : 7.000000
The average turnaround time : 13.200000

---------- SHORTEST JOB REMAINING SCHEDULING ---------
```

ANALYSIS of shortest job remaining scheduling

SRJF is a scheduling method that is a pre-emptive version of shortest job next scheduling.

The process with the smallest amount of time remaining until completion is selected to execute.

Since the currently executing process is the one with the shortest amount of time remaining by definition, a processes will always run until they complete or a new process is added that requires a smaller amount of time.

Figure 7: (On selecting option 5 form menu) ROUND ROBIN scheduling Gantt chart and execution time

```
--------------- ROUND ROBIN SCHEDULING ---------------


ENTER QUANTUM VALUE : 2

THE ESTABLISHED SEQUENCE OF EXECUTION

-----------------------------------------------
PROCESS         REMAINING TIME  TIME ELAPSED
-----------------------------------------------
P[1]            8               Time : 2
P[2]            6               Time : 4
P[3]            0               Time : 6

COUNTER : 1

P[4]            4               Time : 8
P[5]            3               Time : 10
P[1]            6               Time : 12
P[2]            4               Time : 14
P[4]            2               Time : 16
P[5]            1               Time : 18
P[1]            4               Time : 20
P[2]            2               Time : 22
P[4]            0               Time : 24

COUNTER : 2

P[5]            0               Time : 25

COUNTER : 3

P[1]            2               Time : 27
P[2]            0               Time : 29

COUNTER : 4

P[1]            0               Time : 31

COUNTER : 5
```

```
  ALL PROCESSES HAVE COMPLETED EXECUTION

-----------------------------------------
Process         Waiting         Turnaround
-----------------------------------------
P[1]            21              31
P[2]            21              29
P[3]            4               6
P[4]            18              24
P[5]            20              25


The average waiting time in RR : 16.799999
The average turnaround time in RR : 23.000000

--------------- ROUND ROBIN SCHEDULING ---------------
```

ANALYSIS of round robin scheduling

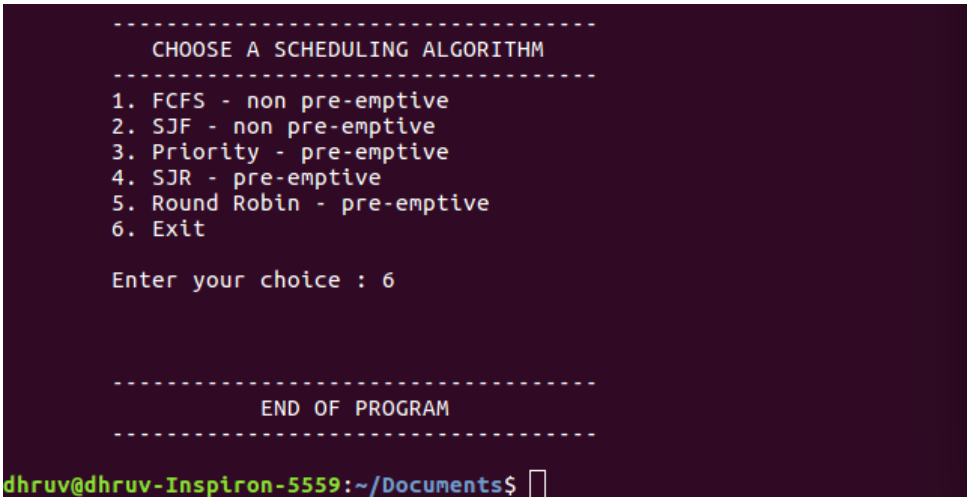
Round Robin is a CPU scheduling algorithm where each process is assigned a fixed time slot in a cyclic way, handling all processes without priority (also known as cyclic execution).

It is starvation-free as all processes get fair share of CPU.

It is pre-emptive as processes are assigned CPU only for a fixed slice of time at most.

The disadvantage of it is more overhead of context switching.

Figure 8: (On selecting option 6 form menu) Program terminates



OVERALL SCHEDULING ALGORITHMS ANALYSIS

Algorithm	Average witing time (s)	Average turnaround time (s)
First come first serve	14.8	21
Shortest job first	8.6	14.8
Priority	11.4	17.6
Shortest job remaining	7	13.2
Round robin	16.8	23

Performance: **Round robin (WORST)** < First come first serve < Priority < Shortest job first < **Shortest job remaining first (BEST)**

For the Xerox system, we can see that the algorithm that favour the

(i) **Orders:** *small turnaround time for photocopying means that the order gets completed quickly*
(ii) **Customers:** *small waiting time for customer means that the customer's work gets completed quickly*

Thus, the algorithm that favours both – the orders and customers is the "shortest job remaining first" pre-emptive scheduling algorithm.