PROGRAMMING ASSIGNMENT

Course code: CSE316

Course Title: Operating System

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

NAME : HemaSundar.T

Reg No: 11716255

Section: K17AP

Roll No : B42

Question Assigned: 2

Submitted to : Baljit Singh Saini sir.

CONTENTS

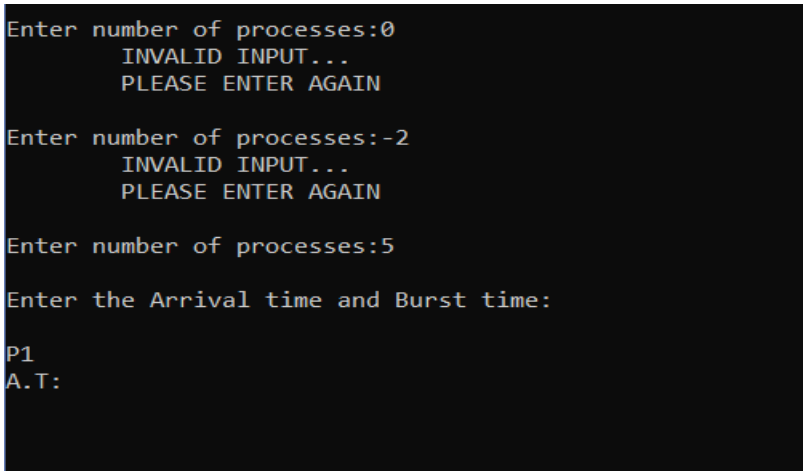
1. PROGRAM DECRIPTION :

Consider a scheduling approach which is non pre-emptive similar to shortest job next in nature. The priority of each job is dependent on its estimated run time, and also the amount of time it has spent waiting. Jobs gain higher priority the longer they wait, which prevents indefinite postponement. The jobs that have spent a long time waiting compete against those estimated to have short run times. The priority can be computed as : Priority = 1+ Waiting time / Estimated run time Write a program to implement such an algorithm.

2. TEST CASES :

a. TAKING WRONG INPUTS

If Number of Processes are less than or equal to zero



If Input for Arrival Time is less than zero and If Input for burst Time is less than or equal to zero.

```
Enter the Arrival time and Burst time:

P1
A.T:-2
        INVALID INPUT...
        PLEASE ENTER AGAIN
A.T:0
B.T:0
        INVALID INPUT...
        PLEASE ENTER AGAIN
B.T:-5
        INVALID INPUT...
        PLEASE ENTER AGAIN
B.T:3
P2
A.T:
```

b. TAKING SAME ARRIVAL TIME FOR DIFFERENT PROCESSES

Input: P1,P4 have same arrival time

```
    Process       Arrival Time        Burst Time
      P1              0                   2
      P2              2                   1
      P3              9                   3
      P4              0                   6
```

Output: Order of Input is preferred

```
ORDER OF EXECUTION BY SJF:

Process  Arrival Time   Burst Time      Start Time      End Time
   P1          0          2               0              2
   P2          2          1               2              3
   P4          0          6               3              9
   P3          9          3               9              12
```

c. NO PROCESS IS RUN FOR A PARTICULAR TIME

Input:

```
Process       Arrival Time        Burst Time
  P1              2                   1
  P2              0                   2
  P3              3                   7
  P4              11                  9
```

Output:

P3 ends at 10 and P4 starts at 11,
No process runs between time 10-11

```
ORDER OF EXECUTION BY SJF:

Process  Arrival Time   Burst Time      Start Time      End Time
   P2          0          2               0              2
   P1          2          1               2              3
   P3          3          7               3              10
   P4          11         9               11             20
```

3. EXAMPLE :

INPUT

```
Process       Arrival Time        Burst Time
  P1              3                   2
  P2              2                   1
  P3              1                   4
  P4              0                   7
  P5              6                   3
```

NON-PREEMPTIVE SJF

Processes are run by SJF(non-pre-emptive) Algorithm and Hence, Calculating waiting time, turn-around time for each process and on average for all processes.

```
ORDER OF EXECUTION BY SJF:

Process  Arrival Time   Burst Time      Start Time      End Time        Waiting Time    Turn Around Time   Priority
   P4          0          7               0               7               0                7             1.00
   P2          2          1               7               8               5                6             6.00
   P1          3          2               8               10              5                7             3.50
   P5          6          3               10              13              4                7             2.33
   P3          1          4               13              17              12               16            4.00

       AVERAGE WAITING TIME IS 5.20
       AVERAGE TURN AROUND TIME IS 8.60
```

EXECUTION BY PRIORITY

Processes are run by Priority Scheduling(non-pre-emptive) Algorithm from the priorities obtained from formula, Priority = 1+ Waiting time / Estimated run time. Hence, Calculating waiting time, turn-around time for each process and on average for all processes.

```
ORDER OF EXECUTION BY PRIORITY:

Process  Arrival Time   Burst Time      Start Time      End Time        Waiting Time    Turn Around Time   Priority
   P4          0          7               0               7               0                7             1.00
   P5          6          3               7               10              1                4             2.33
   P1          3          2               10              12              7                9             3.50
   P3          1          4               12              16              11               15            4.00
   P2          2          1               16              17              14               15            6.00

       AVERAGE WAITING TIME IS 6.60
       AVERAGE TURN AROUND TIME IS 10.00
```

4. GIT HUB LINK :

https://github.com/HemaSundar53/Non-Preemptive-SJF_Priority

5. CODE :

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

struct process{
   int arrival;
   int burst;
   int waiting;
   int turn_around;

   int p_no;
   int start_tym;
   int end_tym;
   float priority;
   }p[80];

void print()
{   printf("\tINVALID INPUT...\n\tPLEASE ENTER AGAIN\n");
}

int main()
{
   // INPUT FOR NUMBER OF PROCESSES
   int n;
   redo1:
   printf("\nEnter number of processes:");
   scanf("%d",&n);
   if(n<=0)
   {
      print();
```

```
        goto redo1;
    }

    // INPUT FOR ARRIVAL AND BURST TIME
    printf("\nEnter the Arrival time and Burst time:\n\n");
    for(int i=0;i<n;i++)
    {
        p[i].p_no=i+1;

        printf("P%d\n",p[i].p_no);
        redo2:
        printf("A.T:");
        scanf("%d",&p[i].arrival);
        if(p[i].arrival<0)
        {
            print();
            goto redo2;
        }
        redo3:
        printf("B.T:");
        scanf("%d",&p[i].burst);
        if(p[i].burst<=0)
        {
            print();
            goto redo3;
        }
    }

    // GIVEN INFORMATION
    printf("\n\nGiven information is:\n");
    printf("\n Process Arrival Time Burst Time\n");
```

```
   for(int i=0;i<n;i++)
   {
      printf("\tP%d\t\t%d\t\t
%d\n",p[i].p_no,p[i].arrival,p[i].burst);
   }

   // SORTING PROCESSES ACCORDING TO
ARRIVAL TIME
   int temp;
   for(int j=0;j<n-1;j++)
   {
      for(int i=0;i<n-j-1;i++)
      {
         if(p[i].arrival>p[i+1].arrival)
            {
               //PROCESS ID
               temp=p[i].p_no;
               p[i].p_no=p[i+1].p_no;
               p[i+1].p_no=temp;
               //ARRIVAL TIME SORT
               temp=p[i].arrival;
               p[i].arrival=p[i+1].arrival;
               p[i+1].arrival=temp;
               // BURST TIME SORT
               temp=p[i].burst;
               p[i].burst=p[i+1].burst;
               p[i+1].burst=temp;
            }
      }
   }

   // SJF ALGORITHM
   int min,burst_compare=0,x=1;
```

```
   for(int i=0;i<n;i++)
   {
      burst_compare=burst_compare+p[i].burst;
      min=p[x].burst;
      for(int j=x;j<n;j++)
      {

if(p[j].arrival<=burst_compare&&min>p[j].burst)
         {
            temp=p[x].p_no;
            p[x].p_no=p[j].p_no;
            p[j].p_no=temp;
            temp=p[x].arrival;
            p[x].arrival=p[j].arrival;
            p[j].arrival=temp;
            min=p[j].burst;
            temp=p[x].burst;
            p[x].burst=p[j].burst;
            p[j].burst=temp;
         }
      }
      x++;
   }

   // CALCULATION OF START TIME, END TIME
AND WAITING TIME, TURN AROUND TIME
   for(int i=0;i<n;i++)
   {
      if(i==0 || p[i].arrival>p[i-1].end_tym)
      {
         p[i].start_tym=p[i].arrival;
      }
      else
```

```
        {
           p[i].start_tym=p[i-1].end_tym;
        }
        p[i].end_tym=p[i].start_tym+p[i].burst;
        p[i].waiting=p[i].start_tym-p[i].arrival;
        p[i].turn_around=p[i].waiting+p[i].burst;
       // p[i].turn_around=p[i].waiting+p[i].burst;
    }

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

p[i].priority=1+((float)p[i].waiting/(float)p[i].burst);
    }

    // PRINTING SJF INFO BEFORE PRIORITY
SORTING
    printf("\n\n\nORDER OF EXECUTION BY
SJF:\n");
    printf("\nProcess\t Arrival Time\tBurst
Time\tStart Time\tEnd Time\tWaiting Time\tTurn
Around Time   Priority\n");
    for(int i=0;i<n;i++)
    {
       printf(" P%d\t\t%d\t %d\t\t %d\t\t %d\t\t
%d\t\t\t%d\t\t
%.2f\n",p[i].p_no,p[i].arrival,p[i].burst,p[i].start_tym
,p[i].end_tym,p[i].waiting,p[i].turn_around,p[i].priori
ty);
    }

    float ex1=0,ey1=0;
```

```
   for(int i=0;i<n;i++)
   {
      ex1=(float)p[i].waiting+ex1;
      ey1=(float)p[i].turn_around+ey1;
   }
   printf("\n\tAVERAGE WAITING TIME IS
%.2f\n\tAVERAGE TURN AROUND TIME IS
%.2f\n",ex1/n,ey1/n);


   // SORTING BY PRIORITY
   burst_compare=0;
   float min_x;
   float temp_x;
   int y=1;
   for(int i=0;i<n;i++)
   {
      burst_compare=burst_compare+p[i].burst;
      min_x=p[y].priority;
      for(int j=y;j<n;j++)
      {

if(p[j].arrival<=burst_compare&&min_x>p[j].priorit
y)
         {
            temp=p[y].p_no;
            p[y].p_no=p[j].p_no;
            p[j].p_no=temp;
            temp=p[y].arrival;
            p[y].arrival=p[j].arrival;
            p[j].arrival=temp;
            temp=p[y].burst;
            p[y].burst=p[j].burst;
```

```
            p[j].burst=temp;
            min_x=p[j].priority;
            temp_x=p[y].priority;
            p[y].priority=p[j].priority;
            p[j].priority=temp_x;
         }
      }
      y++;
   }

   // CALCULATION OF START TIME, END TIME AND WAITING TIME @ priority sorting
   for(int i=0;i<n;i++)
   {
      if(i==0 || p[i].arrival>p[i-1].end_tym)
      {
         p[i].start_tym=p[i].arrival;
      }
      else
      {
         p[i].start_tym=p[i-1].end_tym;
      }
      p[i].end_tym=p[i].start_tym+p[i].burst;
      p[i].waiting=p[i].start_tym-p[i].arrival;
      p[i].turn_around=p[i].waiting+p[i].burst;
   }

   // FINAL SJF INFO AFTER PRIORITY SORTING
   printf("\n\n\nORDER OF EXECUTION BY PRIORITY:\n");
```

```
   printf("\nProcess\t Arrival Time\tBurst
Time\tStart Time\tEnd Time\tWaiting Time\tTurn
Around Time   Priority\n");
   for(int i=0;i<n;i++)
   {
      printf(" P%d\t\t%d\t %d\t\t %d\t\t %d\t\t
%d\t\t\t%d\t\t
%.2f\n",p[i].p_no,p[i].arrival,p[i].burst,p[i].start_tym
,p[i].end_tym,p[i].waiting,p[i].turn_around,p[i].priori
ty);
   }

   float ex2=0,ey2=0;
   for(int i=0;i<n;i++)
   {
      ex2=(float)p[i].waiting+ex2;
      ey2=(float)p[i].turn_around+ey2;
   }
   printf("\n\tAVERAGE WAITING TIME IS
%.2f\n\tAVERAGE TURN AROUND TIME IS
%.2f\n",ex2/n,ey2/n);

   return 0;
}
```

----------X----------

THANK YOU