Name: V.P.Abishek Roll no: 205001002 Ex3- Implementation of CPU Scheduling Policies: FCFS and SJF

Q)Develop a menu driven C program to implement the CPU Scheduling Algorithms FCFSand SJF

Aim :

To develop a menu driven C program to implement the CPU Scheduling Algorithms FCFS and SJF.

Algorithm : (FCFS)

1: Input the number of processes from the user.

2: Have a structure with pid, waiting, burst, arrival, turn_around and completion as data members.

3: Using a loop, input all the given details for each process and store it.

4: Sort the array based on the arrival time of each process.

5: Have 2 variables, average_weight and avg_ta both initialized to 0.

6: Initialize the completion time and turn_around of 1st process as burst of the 1st process.

7: Using a loop, from 1 to n,

7.1: If the completion time of previous process is greater than or equal to arrival time of current process,

7.1.1: Assign completion time of that process as sum of completion time of previous and burst of current process.

7.2 : Else

7.2.1: completion time of that process is the sum of the burst time and arrival time of that process.

7.3: Waiting time of current process is equal to the difference between the completion time and sum of arrival and burst of that current process.

7.4: average weight is incremented by the waiting time of that process.

7.5 : turn_around of that process is the difference between the completion time and arrival time of that process.

7.6 : average_ta is incremented by the turn_around of that process.

8: Print all the details of the processes. Also print the gantt chart along with the average weighting time and average turn_around time.

Algorithm : (SJF)

1: Input the number of processes from the user.

2: Have a structure with pid, waiting, burst, arrival, turn_around and completion as data members.

3: Using a loop, input all the given details for each process and store it.

4: Sort the array based on the burst of each process.

5: Have 2 variables, average_weight and avg_ta both initialized to 0.

6: Initialize the completion time and turn_around of 1st process as burst of the 1st process.

7: Using a loop, from 1 to n,

7.1: Assign waiting time of that process as completion time of the previous process.

7.2: Assign completion time of that process with the sum of the completion time of previous process and burst time of current process.

7.3: Assign turn_around of current process as the completion time of that process.

7.4: Increment the average_weight by the waiting time of the current process.

7.5 Increment the average_ta by the turn_around time of current process.

8: Print all the details of the processes.

9: Print the gantt chart and also print the average waiting and turn time.

Source code:

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

typedef struct schedule * SCH;
typedef struct schedule
{
   char process[3];
   int waiting;
   int arrival;
   int turn_around;
   int burst;
}sch;

void sortarrival(SCH a[], int n)
{
   for (int i = 0; i < n; i++)
   {
        for (int j = i + 1; j < n; j++)
        {
        if (a[i] -> arrival > a[j] -> arrival)
```

```
        {
        SCH temp;
        temp = a[i];
        a[i] = a[j];
        a[j] = temp;
        }
        }
    }
}

void sortburst(SCH a[], int n)
{
    for (int i = 0; i < n; i++)
    {
        for (int j = i + 1; j < n; j++)
        {
        if (a[i] -> burst > a[j] -> burst)
        {
        SCH temp;
        temp = a[i];
        a[i] = a[j];
        a[j] = temp;
        }
        }
    }
}

void gantt_chart(SCH a[], int n)
{
    int i, j;
    printf(" ");
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < a[i] -> burst; j++)
        printf("--");

        printf("- ");
    }
    printf("\n| ");

    for (i = 0; i < n; i++)
    {
        for (j = 0; j < a[i] -> burst - 1; j++)
        printf(" ");

        printf("%s", a[i] -> process);
        for (j = 0; j < a[i] -> burst; j++)
        printf(" ");
```

```
        printf("\b");
        printf("| ");
    }
    printf("\n ");

    for (i = 0; i < n; i++)
    {
        for (j = 0; j < a[i] -> burst; j++)
        printf("--");

        printf("- ");
    }
    printf("\n");
    printf("0");
    int comp = a[0] -> turn_around;
    for (i = 0; i < n - 1; i++)
    {
        for (j = 0; j < a[i] -> burst; j++)
        printf(" ");
        printf(" ");
        if (comp > 9)
        printf("\b");

        printf("%d", comp);
        comp += a[i + 1] -> burst;
    }
    printf(" %d", comp);
    printf("\n");
}

int main()
{
    int n;
    char ch;
    do
    {
        printf("Menu:\n1.FCFS\n2.SJF\n");
        int choice;
        scanf("%d", & choice);
        printf("Enter number of processes: ");
        scanf("%d", & n);
        if (choice == 1)
        {
        printf("--------------FCFS Scheduler-------------------\n");
        SCH a[n];
        for (int i = 0; i < n; i++)
        {
```

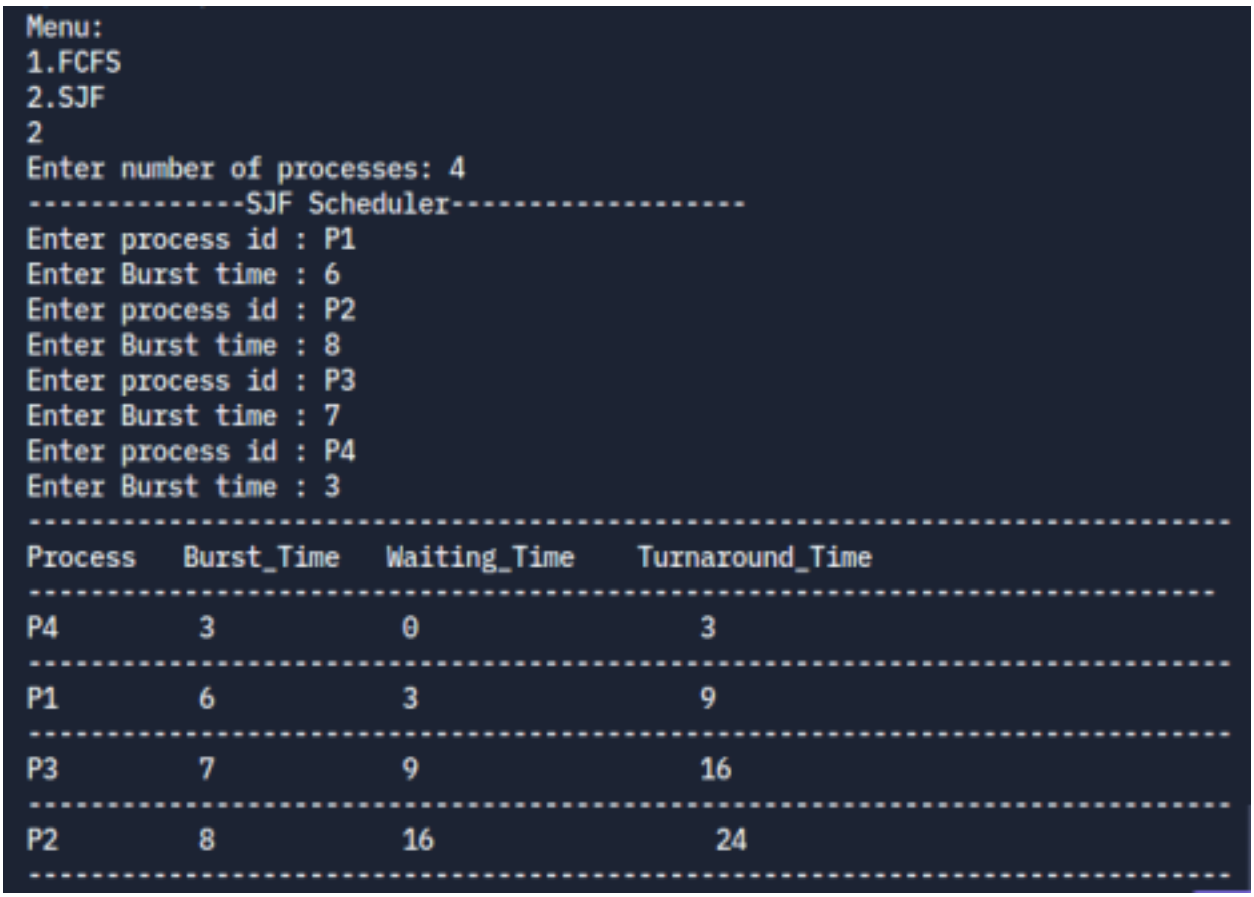
```
a[i] = malloc(sizeof(sch));
printf("Enter process id : ");
scanf("%s", a[i] -> process);
printf("Enter arrival time : ");
scanf("%d", & a[i] -> arrival);
printf("Enter Burst time : ");
scanf("%d", & a[i] -> burst);
}
sortarrival(a, n);
double avg = 0, turn = a[0] -> burst, sum_burst = a[0] -> burst;
a[0] -> waiting = 0;
a[0] -> turn_around = a[0] -> burst;
for (int i = 1; i < n; i++)
{
a[i] -> waiting = sum_burst - a[i] -> arrival;
if (a[i] -> waiting < 0) a[i] -> waiting = 0;

 a[i] -> turn_around = a[i] -> waiting + a[i] -> burst;

avg += a[i] -> waiting;
turn += a[i] -> turn_around;
sum_burst += a[i] -> burst;
}


printf("-------------------------------------------------------------------
-----------------------\n");
      printf("Process Arrival_Time Burst_Time Waiting_Time
Turnaround_Time\n");

printf("-------------------------------------------------------------------
-----------------------\n");
      for (int i = 0; i < n; i++)
      {
      printf("%s %d %d %d
       %d\n", a[i] -> process, a[i] -> arrival, a[i] -> burst, a[i] ->
waiting, a[i] -> turn_around);

printf("-------------------------------------------------------------------
-----------------------\n");
      }
      printf("\n");
      printf("Average Waiting time : %.2f\n", avg / n);
       printf("Average Turn_around time : %.2f\n", turn / n);
      gantt_chart(a, n);
      }
      else if (choice == 2)
      {
```

```
        SCH a[n];
        printf("--------------SJF Scheduler-------------------\n");
        for (int i = 0; i < n; i++)
        {
        a[i] = malloc(sizeof(sch));
        printf("Enter process id : ");
        scanf("%s", a[i] -> process);
        printf("Enter Burst time : ");
        scanf("%d", & a[i] -> burst);
        }
        sortburst(a, n);
        double avg = 0, turn = a[0] -> burst, sum_burst = a[0] -> burst;
        a[0] -> waiting = 0;
        a[0] -> turn_around = a[0] -> burst;
        for (int i = 1; i < n; i++)
        {
        a[i] -> waiting = sum_burst;
         a[i] -> turn_around = a[i] -> waiting + a[i] -> burst;

        avg += a[i] -> waiting;
        turn += a[i] -> turn_around;
        sum_burst += a[i] -> burst;
        }


printf("------------------------------------------------------------------
----------\n");
                printf("Process Burst_Time Waiting_Time Turnaround_Time\n");

printf("------------------------------------------------------------------
---------\n");
        for (int i = 0; i < n; i++)
        {
        printf("%s %d %d %d\n", a[i] -> process, a[i] -> burst, a[i]
-> waiting, a[i] -> turn_around);

printf("------------------------------------------------------------------
----------\n");
        }
        printf("\n");
        printf("Average Waiting time : %.2f\n", avg / n);
         printf("Average Turn_around time : %.2f\n", turn / n);
        gantt_chart(a, n);

        }
        printf("Do you want to continue(Y/N) : ");
        scanf("%s", & ch);
    } while (ch == 'Y');
```

```
}
```

Output:

```
~/OS-lab$ ./ex3
Menu:
1.FCFS
2.SJF
1
Enter number of processes: 3
-------------FCFS Scheduler------------------
Enter process id : P1
Enter arrival time : 0
Enter Burst time : 12
Enter process id : P2
Enter arrival time : 0
Enter Burst time : 7
Enter process id : P3
Enter arrival time : 0
Enter Burst time : 3
---------------------------------------------------------------------------------------
Process   Arrival_Time   Burst_Time   Waiting_Time   Turnaround_Time
---------------------------------------------------------------------------------------
P1        0             12           0              12
---------------------------------------------------------------------------------------
P2        0             7            12             19
---------------------------------------------------------------------------------------
P3        0             3            19             22
---------------------------------------------------------------------------------------

Average Waiting time :  10.33
Average Turn_around time : 17.67
 ------------------------ -------------- -------
|           P1           |       P2     |   P3  |
 ------------------------ -------------- -------
0                        12             19      22
Do you want to continue(Y/N) : Y
```



Learning Outcome :

- Learnt to implement a menu driven C program for CPU scheduling algorithms FCFS and SJF.
- Printed Gantt chart for all the processes.
- Understood the implementation and working of those algorithms.