SSN COLLEGE OF ENGINEERING, KALAVAKKAM (An Autonomous Institution, Affiliated to Anna University, Chennai) DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING UCS1411 - OPERATING SYSTEMS LAB

Lab Exercise 3: Implementation of CPU Scheduling Policies: FCFS and SJF (Non-preemptive and Preemptive)

PROGRAM:

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
typedef struct
       char pid[5];
       float at,bt,st,ft,tat,wt;
}process;
process * input(int n)
       static process p[100];
       //process *p=malloc(sizeof(process)*n);
       for(int i=0;i< n;i++)
              printf("\n\t\tProcess ID :
                                          ");
              scanf("%s",p[i].pid);
              printf("\t\tArrival Time:
                                          ");
             scanf("%f",&p[i].at);
              printf("\t\Burst Time :
                                          ");
             scanf("%f",&p[i].bt);
       return p;
void ganttchart(process p[100],int n)
       int i;
       printf("\nGANTT CHART\n\n");
       for( i=0; i< n; i++)
              printf("+------;
       printf("\n");
```

```
for( i=0; i< n; i++)
                 printf("| %2s |",p[i].pid);
     printf("\n");
      for( i=0; i< n; i++)
           printf("+------;);
     printf("\n");
      for( i=0; i< n; i++)
           printf("%-.1f ",p[i-1].ft);
     printf("%-.1f",p[i-1].ft);
     printf("\n\n");
}
void output(process p[100], int n)
printf("\n-----
----\n'');
      printf("Process ID\tArrival time\tBurst time\tTurnaround time\tWaiting time\n");
printf("-----
----\n'');
      for (int i = 0; i < n; i++)
printf("%s\t\t%.2f\t\t%.2f\t\t%.2f\t\t%.2f\n\n",p[i].pid,p[i].at,p[i].bt,p[i].st,p[i].ft,p[i].tat,p[i].
wt);
}
void fcfs(process p[100], int n)
      process temp;
      for(int i=0;i< n;i++)
           for(int j=i+1;j< n;j++)
                 if(p[i].at>p[j].at)
                       temp=p[i];
                       p[i]=p[j];
                       p[j]=temp;
      float tot wt=0, tot tat=p[0].bt;
      p[0].wt=0;
     p[0].tat=p[0].bt;
      p[0].st=0;
```

```
p[0].ft=p[0].bt;
      for(int i=1;i< n;i++)
             p[i].st = p[i-1].ft;
             p[i].ft = p[i].st + p[i].bt;
             p[i].tat = p[i].ft - p[i].at;
             p[i].wt = p[i].tat - p[i].bt;
             tot_tat += p[i].tat;
             tot wt += p[i].wt;
       }
      ganttchart(p,n);
      output(p,n);
      printf("\t\t\t\t\t\t\t\t\t\t\t\----\t\----\n");
      printf("\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\----\t\----\n");
}
void sjf(process p[100],int n)
      printf("\n\n\t\tNON PREEMPTIVE SJF:\n\n");
      float tot tat=0,tot wt=0;
      int completed=0,lap=0,min=9999,small=0,done[n];
      int flag=0,st=0,ft=0;
      process temp[100];
      //process *temp=malloc(sizeof(process)*n);
      for(int i=0;i< n;++i)
      done[i]=0;
      while(completed!=n)
             for(int i=0;i< n;++i)
                    if(p[i].at \le lap \&\& p[i].bt \le min \&\& done[i]!=1)
                           min=p[i].bt;
                           small=i;
                           flag=1;
                    }
             }
             lap+=p[small].bt;
             st=ft;
             ft+=p[small].bt;
             p[small].st=st;
             p[small].ft=ft;
```

```
p[small].tat=p[small].ft-p[small].at;
             p[small].wt=p[small].tat-p[small].bt;
             tot tat+=p[small].tat;
             tot wt+=p[small].wt;
             done[small]=1;
             temp[completed]=p[small];
             completed++;
             min=9999;
       }
      ganttchart(temp,n);
      output(p,n);
      printf("\t\t\t\t\t\t\t\t\t\t\t\t\t\----\t\----\n");
      printf("\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\----\t\----\n");
}
void presif(process p[100],int n)
      printf("\n\n\t\tPREEMPTIVE SJF:\n\n");
      float rt[n],tot tat=0,tot wt=0;
      int completed=0,lap=0,min=9999,small=0;
      int flag=0,k=0;
      float time[50];
      char id[50][10];
      for (int i = 0; i < n; ++i)
             rt[i]=p[i].bt;
      while(completed!=n)
             for (int i = 0; i < n; ++i)
                    if((p[i].at \le lap) && (rt[i] \le min) && rt[i] \ge 0)
                           min=rt[i];
                           small=i;
                           flag=1;
                    }
             if(flag==0)
                    if(stremp(id[k],id[k-1]==0))
```

```
strcpy(id[k],id[k-1]);
                       time[k-1]++;
               lap++;
               continue;
        }
       if(k \!\! = \!\! = \!\! 0 \parallel strcmp(id[k], id[k \!\! - \!\! 1])! \!\! = \!\! 0) \{
               strcpy(id[k],p[small].pid);
               time[k]=1;
               k++;
       else
               strcpy(id[k],id[k-1]);
               time[k-1]+=1;
        }
       rt[small]--;
       min=rt[small];
       if(min==0)
               min=9999;
       if(rt[small]==0)
               completed++;
               flag=0;
               p[small].wt=(lap+1)-p[small].bt-p[small].at;
               if(p[small].wt<0)
                       p[small].wt=0;
       lap++;
}
printf("%d",k);
int j=0;
float t_{time}[50]=\{0\};
char t_id[50][10];
for (int i = 0; i < k-1; ++i)
       while(strcmp(id[i],id[i+1])==0)
               i++;
       strcpy(t_id[j],id[i]);
       t_{time[j+1]=i+1};
       j++;
}
```

```
printf("\n\nGANTT CHART: \n");
      for( int i=0;i< j;i++)
            printf("+------;);
      printf("\n");
     for(int i=0;i<j;++i)
            printf("| %2s |",t_id[i]);
     printf("\n");
      for (int i=0; i< j; i++)
            printf("+------;
      printf("\n");
      for(int i=0; i< j+1; ++i)
            printf("\n');
      for (int i = 0; i < n; ++i)
            p[i].tat=p[i].bt+p[i].wt;
            tot wt+=p[i].wt;
            tot tat + = p[i].tat;
      output(p,n);
      printf("\t\t\t\t\t\t\t\t\t\t\t\t\t\----\t\----\n");
     printf("\t\t\t\t\t\t\t\t\t\t\t\t\t\t\----\t\----\n");
int main()
int op;
char c;
process *p;
int n;
printf("\n\t\t\tCPU SCHEDULING ALGORITHMS\n\n");
do
     printf("\n1.FCFS\n2.SJF\n3.Exit\n\nEnter your option: ");
     scanf("%d",&op);
```

```
if(op==1)
              printf("\n\t\tFCFS CPU SCHEDULER\n\n");
             printf("\t\tNo. of processes: ");
             scanf("%d",&n);
             p=input(n);
             fcfs(p,n);
       }
       else if (op==2)
              char ch;
             printf("\n\t\tSJF CPU SCHEDULER\n\n");
             printf("\t\ta.Non Preemptive SJF\n\t\tb.Preemptive SJF\n\n\t\tEnter your
option: ");
             scanf(" %c",&ch);
             printf("\n\t\tNo. of processes: ");
             scanf("%d",&n);
              p=input(n);
             if(ch=='a')
                    sjf(p,n);
             else if(ch=='b')
                    presif(p,n);
       }
       else
             exit(0);
      printf("Want to continue?(y/n): ");
      scanf(" %c",&c);
}while(c=='y');
return 0;
}
```

OUTPUT:

(base) MSMLs-iMac:ex3 msml\$./fcfs_sjf

CPU SCHEDULING ALGORITHMS

1.FCFS 2.SJF 3.Exit

Enter your option: 1 FCFS CPU SCHEDULER

No. of processes: 5

_						
Process ID	:	P1	Arrival Time:	0	Burst Time:	6
Process ID	:	P2	Arrival Time:	1	Burst Time:	2
Process ID	:	P3	Arrival Time:	1	Burst Time:	3
Process ID	:	P4	Arrival Time:	2	Burst Time:	1
Process ID	:	P5	Arrival Time:	2	Burst Time:	2

GANTT CHART

+	++	++	++	++_	+
P1		P2	P3.	P4	P5
+	++	++	++	++-	+
0.0	6.0	8.0	11.0	12.0	14.0

Process ID	Arrival time	Burst time	Turnaround time	Waiting time	
P1	0.00	6.00	6.00	0.00	
P2	1.00	2.00	7.00	5.00	
Р3	1.00	3.00	10.00	7.00	
P4	2.00	1.00	10.00	9.00	
P5	2.00	2.00	12.00	10.00	
		Average	9.00	6.20	

Want to continue?(y/n): y

CPU SCHEDULING ALGORITHMS

1.FCFS

2.SJF

3.Exit

Enter your option: 2

SJF CPU SCHEDULER

a.Non Preemptive SJF b.Preemptive SJF

Enter your option: a

No. of processes: 5

Process ID	:	P1	Arrival Time:	0	Burst Time:	6
Process ID	:	P2	Arrival Time:	1	Burst Time:	2
Process ID	:	P3	Arrival Time:	1	Burst Time:	3
Process ID	:	P4	Arrival Time:	2	Burst Time:	1
Process ID	:	P5	Arrival Time:	2	Burst Time:	2

NON PREEMPTIVE SJF:

GANTT CHART

+		++		++		++		++		+
ĺ	P1		P4		P2		P5		P3	ĺ
+		++		++		++		++		+
0.0		6.0		7.0		9.0		11.0		14.0

Process ID	Arrival time	Burst time	Turnaround time	Waiting time
P1	0.00	6.00	6.00	0.00
P2	1.00	2.00	8.00	6.00
Р3	1.00	3.00	13.00	10.00
P4	2.00	1.00	5.00	4.00
P5	2.00	2.00	9.00	7.00
		Average	8.20	5.40

Want to continue?(y/n): y

CPU SCHEDULING ALGORITHMS

1.FCFS 2.SJF

3.Exit

Enter your option: 2

SJF CPU SCHEDULER

a.Non Preemptive SJF b.Preemptive SJF

Enter your option: b

No. of processes: 5

Process ID:	P1	Arrival Time:	0	Burst Time:	6
Process ID:	P2	Arrival Time:	1	Burst Time:	2
Process ID:	P3	Arrival Time:	1	Burst Time:	3
Process ID:	P4	Arrival Time:	2	Burst Time:	1
Process ID:	P5	Arrival Time:	2	Burst Time:	2

PREEMPTIVE SJF:

GANTT CHART:

+	 ++	 ++		++	 ++	 ++	 +
•		• • • • • • • • • • • • • • • • • • • •	P4				
•							
0.0	1.0	3.0		4.0	6.0	9.0	14.0

Process ID	Arrival time	Burst time	Turnaround time	Waiting time
P1	0.00	6.00	14.00	8.00
P2	1.00	2.00	2.00	0.00
Р3	1.00	3.00	8.00	5.00
P4	2.00	1.00	2.00	1.00
P5	2.00	2.00	4.00	2.00
		Average	6.00	3.20

Want to continue?(y/n):n