Register No: 22L31A05A3 Experiment No: 02 Date: 09/10/2023

S. No	Component	Max. Marks	Marks Secured
1	Preparedness	2	2
2	Viva-Voce	2	2
3	Experiment	3	3
4	Analysis & Record	3	3
	Total	10	10
Date		Signature of the Lab teacher	

Exercise - 2:

AIM: Stimulate the following cpu scheduling algorithms.

a) Round Robin:

A Round Robin is a cpu scheduling algorithm that shares equal portions of resources in circular orders to each process and handles all process without prioritization. In round robin, each process gets a fixed time interval for execution called quantum time or time slice.

```
#include<stdio.h>
#include<conio.h>
 void main()
{ int i, j, n, m, sum=0, count=0, y, q, wt, tat,
          at[10], bt[10], temp[10];
 float avg-wt, avg-tat;
 printf("Enter no of process");
  scanf("%d"& n);
      y=n;
 for(i=0; i<n; i++)
 { printf("Enter arrival & burst time of process
          P[%d]", i+1);
  scanf("%d", & at[i]);
  scanf("%d", & bt[i]);
```

Register No : Experiment No : Date:

```
        temp[i] = bt[i];
    }
    printf("Enter quantum time\n");
    scanf("%d",&q);
    printf("\n process \t Burst time \t Arrival time \t
        TAT \t waiting time");
    for(sum=0, i=0; y!=0)
    {
        if(temp[i]<=q && temp[i]>0)
        {
            sum = sum+temp[i];
            temp[i]=0;
            count=1;
        }
        else if(temp[i]>0){
            temp[i] = temp[i]-q;
            sum = sum+q;
        }
        if(temp[i]==0 && count==1)
        { y--;
        printf("\n process[%d]\t %d\t %d\t %d\t %d
        %d", i+1, bt[i], sum-at[i], at[i] sum-at[i]
                                    - bt[i]);
        wt = wt+sum-at[i]-bt[i];
        tat = tat+sum-at[i];
            count=0;
        }
        if(i==n-1) i=0;
        else if(at[i+1]<=sum) i++;
            else i=0;
    }
```

Output: Enter no of process: 4

Enter arrival & burst time of process[1] 0 8

Enter arrival & burst time of process[2] 1 5

Enter arrival & burst time of process[3] 2 10

Enter arrival & burst time of process[4] 3 11

Enter quantum time 6

process	Burst time	Arrival time	T.A.T	waiting Time
process[1]	8	0	10	5
process[2]	5	1	25	17
process[3]	10	2	27	17
process[4]	11	3	31	20

Average Turn around time = 14.75

Average waiting time = 23.2500

Register No : | Experiment No : | Date:

```
    avg_wt = wt * 1/n;
    avg_tat = tat * 1/n;
  printf("\n Average Turn around time = %f", avg_wt);
  printf("\n Average waiting time = %f", avg_tat);
    getch();
}
```

b) Shortest Job first (SJF): The shortest job first (SJF) is a scheduling policy that selects the waiting process with the smallest execution time to execute next.

Program:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
  void main()
  { int et[20], at[10], n, i, j, temp, st[10], ft[10], wt[10],
                                ta[10];
    int totwb, totta=0, s, p[10];
    float awt, ata;
    char pn[10][10], t[10];
  printf("Enter no of process");
   scanf("%d", &n);
  printf("Enter burst time and arrival time");
   for(i=0; i<n; i++)
   { scanf("%d%d%d %d", &p[i], &ft[i], &at[i]);
   };
   s = ft[0] + at[0];
      st[0] = s;
  for(i=1; i<n; i++)
  {
```

Register No : Experiment No : Date:

```
for (j=i+1; j<n; j++)
{ if (ft[i]>ft[j])
  { temp = at[i];
    at[i]=at[j];
    at[j]=temp;
   temp=ft[i];
   ft[i]=ft[j];
   ft[j]=temp;
    temp=p[i];
    p[i]=p[j];
    p[j]=temp;
  }
 }
}
for(i=1;i<n;i++)
{ if(s<at[i])
  s=s+(at[i]-s);
   s=s+ft[i];
   st[i]=s;
}
for(i=0;i<n;i++)
 { ta[i]=st[i]-at[i];
   wt[i]=ta[i]-b[i];
 }
for(i=0;i<n;i++)
 { for(j=i+1;j<n;j++)
   { if(p[i]>p[j]){
      temp=at[i];
     at[i]=at[j];
      at[j]=temp;
```

Output:

Enter no of process 5

Enter burst time and arrival time 1 3 0
2 6 2
3 4 4
4 5 6
5 2 8

process	Arrival time	Burst time	completion time	TAT	WT
1	0	3	3	3	0
2	2	6	9	7	1
3	4	4	15	11	7
4	6	5	20	14	9
5	8	2	11	3	1

Average waiting time = 3.6

Average turn around time = 7.6

Register No : Experiment No : Date:

```
            temp = ft[i];
            ft[i] = ft[j];
            ft[j] = temp;
            temp = P[i];
            P[i] = P[j];
            P[j] = temp;
            temp = st[i];
            st[i] = st[j];
            st[j] = temp;
        }
    }
    printf("process\t arrival time\t Burst time\t completion
        time\t TAT\t WT\t\n");
    for (i=0; i<n; i++)
    {
        totwt = totwt + wt[i];
        totta = totta + wta[i];
    }
    for (i=0; i<n; i++)
    {
        printf("%d\t %d\t %d\t %d\t %d\t %d\t\n",
            P[i], at[i], ft[i], st[i], ta[i], wt[i]);
    }
    printf("Average waiting time = %f\n", (float)totwt/n);
    printf("Average turn around time = %f\n", (float)
                                        totta/n);
}
```

6) First Come First Serve: (FCFS)

FCFS scheduling algorithm automatically executes the queued processes and requests in the order of their arrival. It allocates the job that first arrived in the queue to the CPU, then allocates the

Output:

Enter no of process : 5

Enter burst time and Arrival time : 1 3 0
2 6 2
3 4 4
4 5 6
5 2 8

process	Arrival time	Burst time	completion-time	TAT	WT
1	0	3	3	3	0
2	2	6	9	7	1
3	4	4	15	11	7
4	6	5	20	14	9
5	8	2	11	3	1

Average waiting time = 4.6

Average turn around time = 8.6

Register No : Experiment No : Date:

second one and so on.

```
#include <stdio.h>
  int main()
{
 int a[100], b[100], c[100], d[100], e[100], i, j, k, p,
     q, s, sum=0, add=0
  int n;
 printf("Enter no of process");
  scanf("%d", &n);
printf("Enter arrival time");
 scanf("%d" &a[i]);
printf("Enter burst time");
 scanf("%d", &b[i]);
 for(i=0; i<n; i++)
  {
      e[i]=b[i];
      if(i>=1)
      b[i]=b[i]+b[i-1];
      c[i]=b[i]-a[i];
      d[i]=c[i]-e[i];
       sum=sum+c[i];
       add=add+d[i];
  }
printf("\nProcess\t Burst time\t Arrival time\t waiting
       time\t tat");
for(i=0; i<n; i++){
  printf("\n process[%d]\t%d\t%d\t%d\t%d", i+1, e[i], a[i], d[i], c[i]); }
  printf("Average turn around time=");
   printf("%f\n", (float)sum/n);
   printf("Average waiting time=");
   printf("%f\n", (float)add/n);
   return 0;}
```

Output:

Enter number of process : 4

Enter burst time, process name, arrival time, priority

1 0 10 5
2 1 6 4
3 3 2 2
4 5 4 0

Pname	AT	BT	priority	WT	TAT
1	0	10	5	12	22
2	1	6	4	6	12
3	3	2	2	0	2
4	5	4	0	0	4

Average waiting time = 4.5

Average turn around time = 10

Register No : Experiment No : Date:

Process Scheduling Algorithm: In priority scheduling, there is a priority number assigned to each process. In some systems, the lower the number, the higher the priority, while in the others, the higher the number, the higher will be the priority.

Program:

```
#include<stdio.h>
int main()
{ int n, at[10], b[10], st[10], ft[10], wt[10], ta[10],
        et[20], temp;
printf("Enter number of process:");
scanf("%d",&n);
int b[n], p[n], index[n]; char pn[10][10], t[10];
for(int i=0; i<n; i++)
{
printf("Enter Burst time and process name,
    process name and priority");
scanf("%s%d%d%d", pn[i], &at[i], &et[i],
                    &p[i]);
}
int totwt=0, totta=0;
float awt, ata;
for(i=0; i<n; i++)
{ for(j=0; j<n; j++)
  { if(p[i]<p[j])
    { temp=p[i];
      p[i]=p[j];
      p[j]=temp;
      temp=at[i];
      at[i]=at[j];
      at[j]=temp;
      temp=et[i];
```

```
        et[i] = et[j];
        et[j] = temp;
        strcpy(t, pn[j]);
        strcpy(pn[i], pn[j]);
        strcpy(pn[j], t); } }
    for(i=0; i<n; i++)
{   if(i==0){
        st[i] = at[i]; wt[i] = st[i] - at[i]; ft[i] = st[i] + et[i];
        ta[i] = ft[i] - at[i]; }
    else{
        st[i] = ft[i-1]; wt[i] = st[i] - at[i]; ft[i] = st[i] + et[i];
        ta[i] = ft[i] - at[i]; }
        totwt = totwt + wt[i];
        totta = totta + ta[i];
    }
        awt = (float)totwt/n;
        ata = (float)totat/n;
    printf("\n pname\t AT\t BT\t priority\t WT\t TAT\n");
        for(i=0; i<n; i++)
        {
        printf("\n%d\t%d\t%d\t%d\t%d\t%d", pn[i],
            at[i], et[i], p[i], wt[i], ta[i]);
        }
        printf("\n Averagewaiting time=", awt);
        printf("\n Average turn around time=", ata);
    }
```