Harsha Vardhinii.T
230701109

Ex. No.: 6a)

FIRST COME FIRST SERVE

Aim:

To implement First-come First- serve(FCFS) scheduling technique

Program Code:

```
bt=[]
print("Enter the number of process: ")
n=int(input())
print("Enter the burst time of the processes: \n")
bt=list(map(int, input().split()))
wt=[]
avgwt=0
tat=[]
avgtat=0
wt.insert(0,0)
tat.insert(0,bt[0])
for i in range(1,len(bt)):
wt.insert(i,wt[i-1]+bt[i-1])
tat.insert(i,wt[i]+bt[i])
avgwt+=wt[i]
avgtat+=tat[i]
avgwt=float(avgwt)/n
avgtat=float(avgtat)/n
print("\n")
print("Process\t Burst Time\t Waiting Time\t Turnaround Time")
for i in range(0,n):
print(str(i)+"\t\t"+str(bt[i])+"\t\t"+str(wt[i])+"\t
\t"+str(tat[i]))
print("\n")
print("Average Waiting time is: "+str(avgwt))
print("Average Turn Around Time is: "+str(avgtat))
```

Output:

```
Enter the number of process:
3
Enter the burst time of the processes:

24 3 3
Process Burst Time Waiting Time Turn Around Time 0 24 0 24
1 3 24 27
2 3 27 30
Average Waiting time is: 17.0
Average Turn Around Time is: 19.0
```

Ex. No.: 6b)
SHORTEST JOB FIRST
Aim:
To implement the Shortest Job First(SJF) scheduling technique

Program Code:

```
bt=[] #bt stands for burst time
print("Enter the number of process: ")
n=int(input())
processes=[]
for i in range(0,n):
processes.insert(i,i+1)
print("Enter the burst time of the processes: \n")
bt=list(map(int, raw_input().split()))
for i in range(0,len(bt)-1): #applying bubble sort on bt
for j in range(0,len(bt)-i-1):
if(bt[j]>bt[j+1]):
temp=bt[j]
bt[j]=bt[j+1]
bt[j+1]=temp
temp=processes[j]
processes[j]=processes[j+1]
processes[j+1]=temp
wt=[] #wt stands for waiting time
avgwt=0 #average of waiting time
tat=[] #tat stands for turnaround time
avgtat=0 #average of total turnaround time
wt.insert(0,0)
tat.insert(0,bt[0])
for i in range(1,len(bt)):
wt.insert(i,wt[i-1]+bt[i-1])
tat.insert(i,wt[i]+bt[i])
avgwt+=wt[i]
avgtat+=tat[i]
avgwt=float(avgwt)/n
avgtat=float(avgtat)/n
print("\n")

print("Process\t Burst Time\t Waiting Time\t Turn Around Time")
for i in range(0,n):
print(str(processes[i])+"\t\t"+str(bt[i])+"\t\t"+str(wt[i]) +"\t\t"+str(tat[i]))
print("Average Waiting time is: "+str(avgwt)) print("Average Turn Around Time is: "+str(avgtat))
```

Output:

```
Enter the number of process:
4
Enter the burst time of the processes:
8 4 9 5
Process Burst Time Waiting Time Turn Around Time 2 4 0 4
4 5 4 9
1 8 9 17
3 9 17 26
Average Waiting time is: 7.5
Average Turn Around Time is: 13.0
```

Ex. No.: 6c)
PRIORITY SCHEDULING

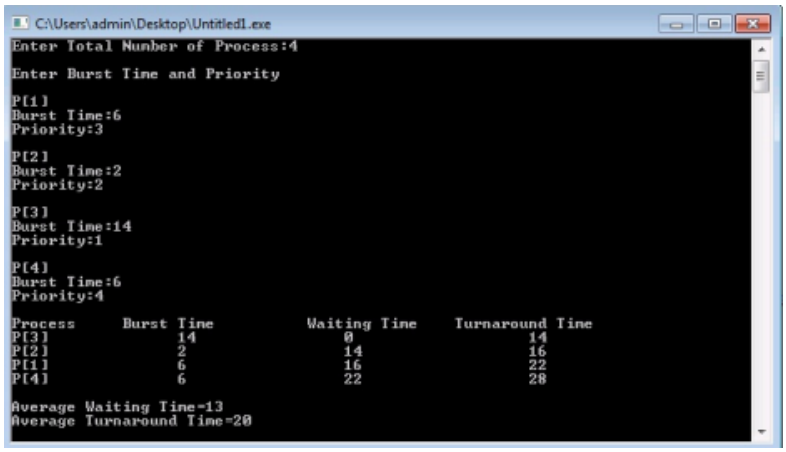
Aim:
To implement priority scheduling technique

Program Code:

```
#include<stdio.h>
int main()
{
int bt[20],p[20],wt[20],tat[20],pr[20],i,j,n,total=0,pos,temp,avg_wt,avg_tat; printf("Enter Total
Number of Process:");
scanf("%d",&n);
printf("\nEnter Burst Time and Priority\n");
for(i=0;i<n;i++)
{
printf("\nP[%d]\n",i+1);
printf("Burst Time:");
scanf("%d",&bt[i]);
printf("Priority:");
scanf("%d",&pr[i]);
p[i]=i+1; //contains process number
}
//sorting burst time, priority and process number in ascending order using selection sort
for(i=0;i<n;i++)
{
pos=i;
for(j=i+1;j<n;j++)
{
if(pr[j]<pr[pos])
pos=j;
}
temp=pr[i];
pr[i]=pr[pos];
pr[pos]=temp;
temp=bt[i];
bt[i]=bt[pos];
bt[pos]=temp;
temp=p[i];
 p[i]=p[pos];
p[pos]=temp;
}
wt[0]=0; //waiting time for first process is zero
//calculate waiting time
for(i=1;i<n;i++)
{
wt[i]=0;
for(j=0;j<i;j++)
wt[i]+=bt[j];
total+=wt[i];
}
avg_wt=total/n; //average waiting time
total=0;
printf("\nProcess\t Burst Time \tWaiting Time\tTurnaround Time"); for(i=0;i<n;i++)
{
```

```
tat[i]=bt[i]+wt[i]; //calculate turnaround time
total+=tat[i];
printf("\nP[%d]\t\t %d\t\t %d\t\t\t%d",p[i],bt[i],wt[i],tat[i]); }
avg_tat=total/n; //average turnaround time
printf("\n\nAverage Waiting Time=%d",avg_wt);
printf("\nAverage Turnaround Time=%d\n",avg_tat);
return 0;
}
```

Output:



Ex. No.: 6d)
ROUND ROBIN SCHEDULING

Aim:
To implement the Round Robin (RR) scheduling technique

Program Code:

```
#include<stdio.h>
int main()
{
int i, limit, total = 0, x, counter = 0, time_quantum;
int wait_time = 0, turnaround_time = 0, arrival_time[10], burst_time[10], temp[10]; float
average_wait_time, average_turnaround_time;
printf("nEnter Total Number of Processes:t");
scanf("%d", &limit);
x = limit;
for(i = 0; i < limit; i++)
{
printf("nEnter Details of Process[%d]n", i + 1);
printf("Arrival Time:t");
scanf("%d", &arrival_time[i]);
printf("Burst Time:t");
 scanf("%d", &burst_time[i]);
temp[i] = burst_time[i];
}
printf("nEnter Time Quantum:t");
scanf("%d", &time_quantum);
printf("nProcess IDttBurst Timet Turnaround Timet Waiting Timen"); for(total = 0, i = 0; x != 0;)
{
if(temp[i] <= time_quantum && temp[i] > 0)
{
total = total + temp[i];
temp[i] = 0;
counter = 1;
}
else if(temp[i] > 0)
{
temp[i] = temp[i] - time_quantum;
total = total + time_quantum;
}
if(temp[i] == 0 && counter == 1)
{
x--;
printf("nProcess[%d]tt%dtt %dttt %d", i + 1, burst_time[i], total - arrival_time[i], total -
arrival_time[i] - burst_time[i]);
wait_time = wait_time + total - arrival_time[i] - burst_time[i]; turnaround_time = turnaround_time +
total - arrival_time[i]; counter = 0;
}
if(i == limit - 1)
{
i = 0;
}
else if(arrival_time[i + 1] <= total)
{
i++;
```

```
}
else
{
i = 0;
}
}
average_wait_time = wait_time * 1.0 / limit;
average_turnaround_time = turnaround_time * 1.0 / limit;
printf("nnAverage Waiting Time:t%f", average_wait_time);
printf("nAvg Turnaround Time:t%fn", average_turnaround_time);
return 0;

}
```

Output: