Name: Sneha Tiwari

Roll: 11500119052 | Group: B1

Assignment 12 - Assignment 2 on Scheduling

**1. Write a program to implement the SJF Scheduling algorithm.**

**a) Non-Preemptive SJF**

**Solution:**

**C Program**

#include<stdio.h>

int main()

{

int bt[20],p[20],wt[20],tat[20],i,j,n,total=0,pos,temp;

float avg\_wt,avg\_tat;

printf("Enter number of processes: ");

scanf("%d",&n);

printf("\nEnter Burst Time:\n");

for(i=0;i<n;i++)

{

printf("P-%d: ",i+1);

scanf("%d",&bt[i]);

p[i]=i+1;

}

for(i=0;i<n;i++)

{

pos=i;

for(j=i+1;j<n;j++)

{

if(bt[j]<bt[pos])

pos=j;

}

temp=bt[i];

bt[i]=bt[pos];

bt[pos]=temp;

temp=p[i];

p[i]=p[pos];

p[pos]=temp;

}

wt[0]=0;

for(i=1;i<n;i++)

{

wt[i]=0;

for(j=0;j<i;j++)

wt[i]+=bt[j];

total+=wt[i];

}

avg\_wt=(float)total/n;

total=0;

printf("\nProcess\t\tBurstTime\tWaitingTime\tTurnaroundTime\n");

for(i=0;i<n;i++)

{

tat[i]=bt[i]+wt[i];

total+=tat[i];

printf("\n\P-%d\t\t %d\t\t %d\t\t\t%d",p[i],bt[i],wt[i],tat[i]);

}

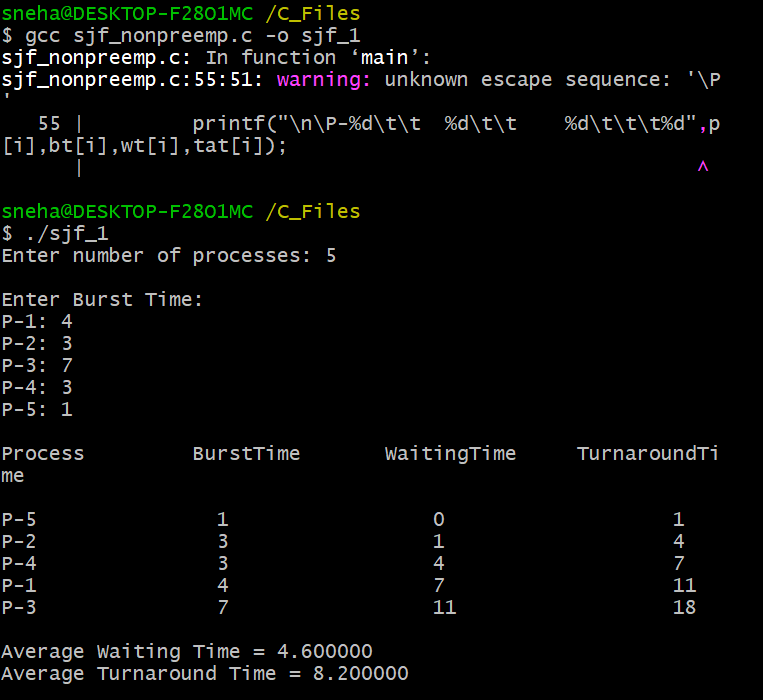
avg\_tat=(float)total/n;

printf("\n\nAverage Waiting Time = %f",avg\_wt);

printf("\nAverage Turnaround Time = %f\n",avg\_tat);

}

**Output:**

****

**b) Preemptive SJF**

**Solution:**

**C Program:**

#include <stdio.h>

int main()

{

int arrival\_time[10], burst\_time[10], temp[10];

int i, smallest, count = 0, time, limit;

double wait\_time = 0, turnaround\_time = 0, end;

float average\_waiting\_time, average\_turnaround\_time;

printf("\nEnter the number of processes: ");

scanf("%d", &limit);

printf("Enter process details: \n", limit);

for(i = 0; i < limit; i++)

{

printf("\nEnter Arrival Time: ");

scanf("%d", &arrival\_time[i]);

printf("Enter Burst Time: ");

scanf("%d", &burst\_time[i]);

temp[i] = burst\_time[i];

}

burst\_time[9] = 9999;

for(time = 0; count != limit; time++)

{

smallest = 9;

for(i = 0; i < limit; i++)

{

if(arrival\_time[i] <= time && burst\_time[i] < burst\_time[smallest] && burst\_time[i] > 0)

{

smallest = i;

}

}

burst\_time[smallest]--;

if(burst\_time[smallest] == 0)

{

count++;

end = time + 1;

wait\_time = wait\_time + end - arrival\_time[smallest] - temp[smallest];

turnaround\_time = turnaround\_time + end - arrival\_time[smallest];

}

}

average\_waiting\_time = wait\_time / limit;

average\_turnaround\_time = turnaround\_time / limit;

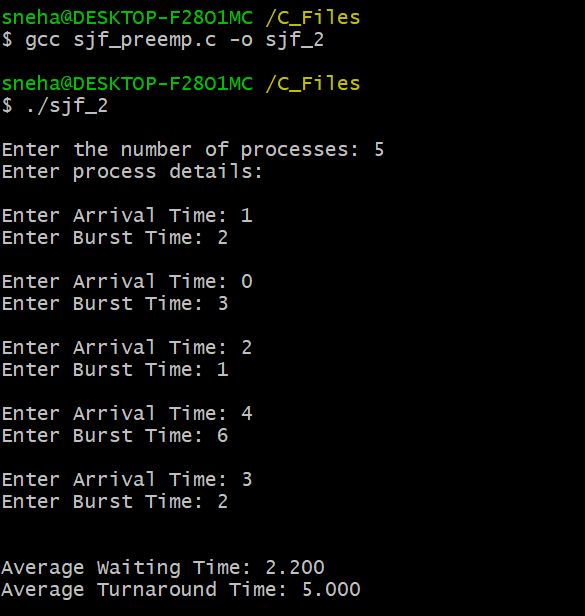
printf("\n\nAverage Waiting Time: %.3f", average\_waiting\_time);

printf("\nAverage Turnaround Time: %.3f\n", average\_turnaround\_time);

return 0;

}

**Output:**



**2. Write a program for implementing the Priority scheduling algorithm.**

**Solution:**

**C Program:**

#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 number of processes: ");

scanf("%d",&n);

printf("Enter Burst Time and Priority:\n");

for(i=0;i<n;i++)

{

printf("\n[P-%d]\n",i+1);

printf("Burst Time: ");

scanf("%d",&bt[i]);

printf("Priority: ");

scanf("%d",&pr[i]);

p[i]=i+1;

}

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;

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;

total=0;

printf("\nProcess\t\tBurstTime\tWaitingTime\tTurnaroundTime\n");

for(i=0;i<n;i++)

{

tat[i]=bt[i]+wt[i];

total+=tat[i];

printf("\n[P-%d] \t\t%d\t\t %d\t\t\t%d",p[i],bt[i],wt[i],tat[i]);

}

avg\_tat=total/n;

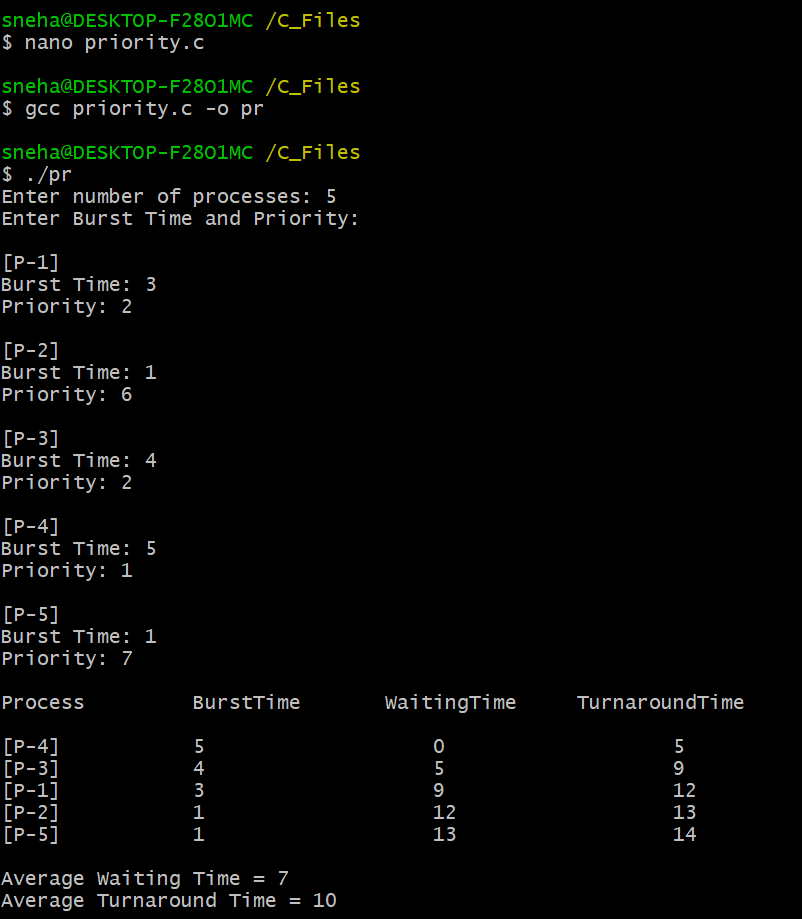
printf("\n\nAverage Waiting Time = %d",avg\_wt);

printf("\nAverage Turnaround Time = %d\n",avg\_tat);

return 0;

}

**Output:**



**3. Write a program for implementing the round robin Scheduling algorithm.**

**Solution:**

**C Program:**

#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 number of processes: ");

scanf("%d", &limit);

printf("\nEnter process details:\n");

x = limit;

for(i = 0; i < limit; i++)

{

printf("\n[P-%d]\n", i + 1);

printf("Arrival Time: ");

scanf("%d", &arrival\_time[i]);

printf("Burst Time: ");

scanf("%d", &burst\_time[i]);

temp[i] = burst\_time[i];

}

printf("\nEnter Time Quantum: ");

scanf("%d", &time\_quantum);

printf("\nProcessID\tBurstTime\tTurnaroundTime\tWaitingTime\n");

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("\n[P-%d] \t\t%d\t\t %d\t\t %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;

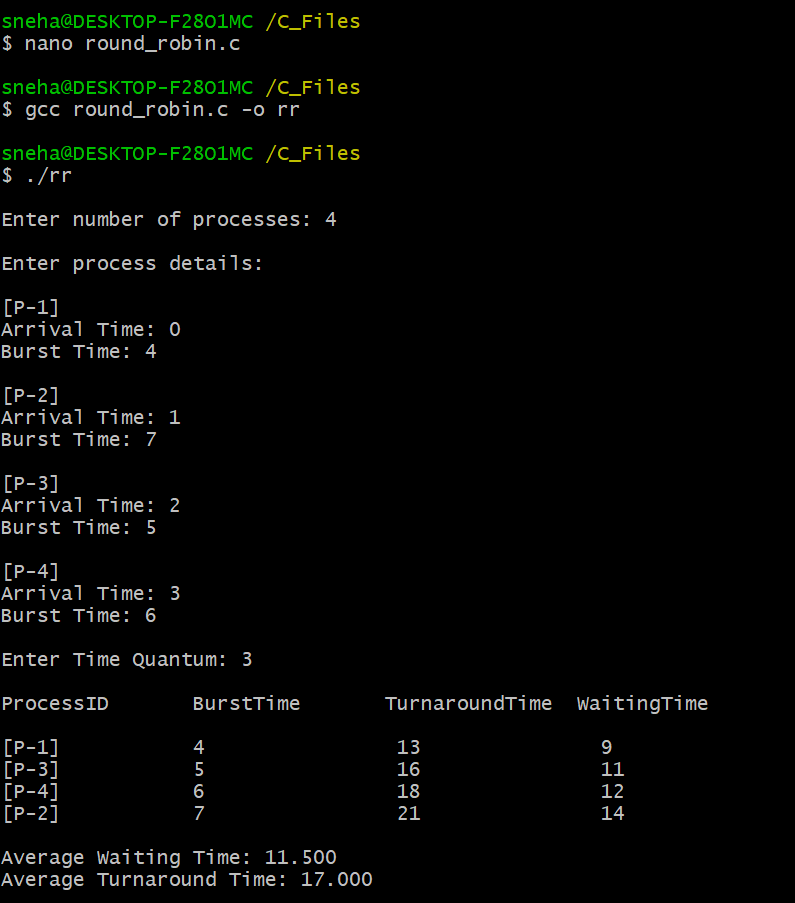
printf("\n\nAverage Waiting Time: %.3f", average\_wait\_time);

printf("\nAverage Turnaround Time: %.3f\n", average\_turnaround\_time);

return 0;

}

**Output:**

****

-----------------------------------------------