EXPERIMENT NO. 3

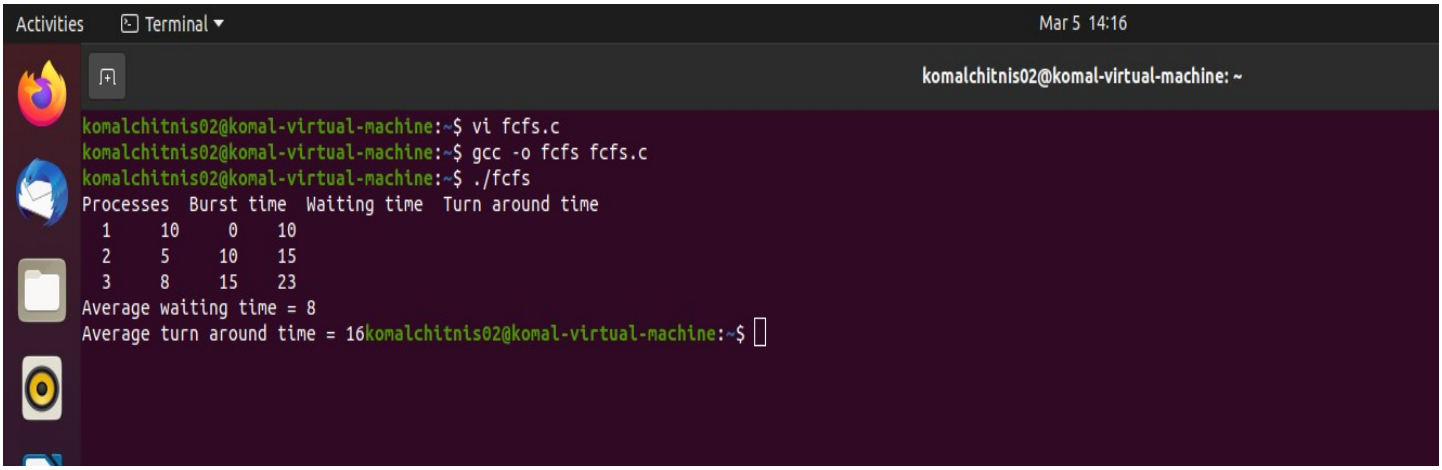
NAME: KOMAL MAHESH CHITNIS
DIV: A
ROLL NO: 26
MOODLE ID: 20102068

FIRST COME FIRST SERVE (FCFS) SCHEDULING IN C PROGRAMMING

```
#include<stdio.h>

void findWaitingTime(int processes[], int n, int bt[], int wt[])
{

	wt[0] = 0;

	for (int i = 1; i < n ; i++ )
		wt[i] = bt[i-1] + wt[i-1] ;
}

void findTurnAroundTime( int processes[], int n, int bt[], int wt[], int tat[])
{

	for (int i = 0; i < n ; i++)
		tat[i] = bt[i] + wt[i];
}


void findavgTime( int processes[], int n, int bt[])
{
	int wt[n], tat[n], total_wt = 0, total_tat = 0;

	findWaitingTime(processes, n, bt, wt);

	findTurnAroundTime(processes, n, bt, wt, tat);

	printf("Processes  Burst time  Waiting time  Turn around time\n");

	for (int i=0; i<n; i++)
	{
		total_wt = total_wt + wt[i];
		total_tat = total_tat + tat[i];
		printf("  %d ",(i+1));
		printf("    %d ", bt[i]);
```

```
            printf("    %d",wt[i]);
            printf("    %d\n",tat[i]);
        }
        int s=(float)total_wt / (float)n;
        int t=(float)total_tat / (float)n;
        printf("Average waiting time = %d",s);
        printf("\n");
        printf("Average turn around time = %d",t);

}

int main()
{
        int processes[] = { 1, 2, 3};
        int n = sizeof processes / sizeof processes[0];

        int burst_time[] = {10, 5, 8};

        findavgTime(processes, n, burst_time);
        return 0;
}
```

OUTPUT:



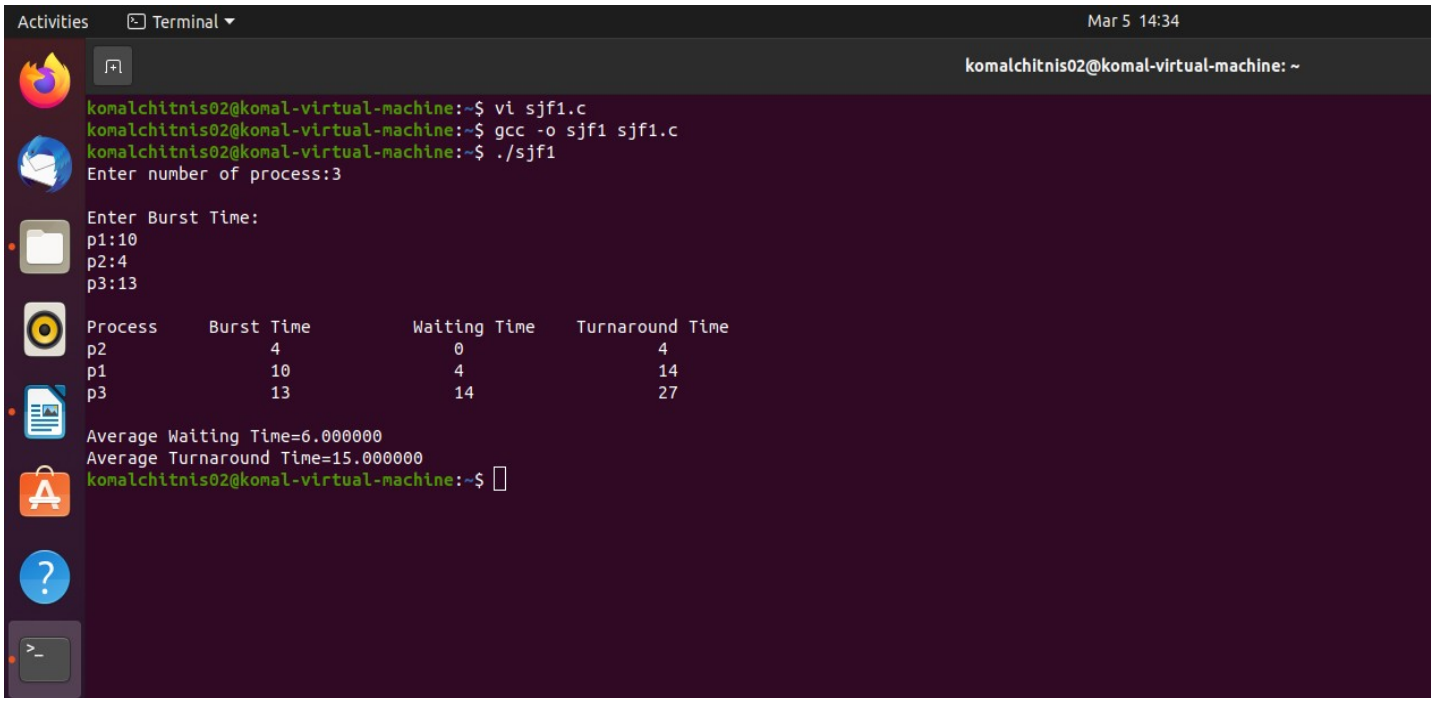
SHORTEST JOB FIRST (SJF) SCHEDULING IN C PROGRAMMING

PRE-EMPTIVE:

```
#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 process:");
   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    Burst Time    \tWaiting Time\tTurnaround Time");
  for(i=0;i<n;i++)
  {
    tat[i]=bt[i]+wt[i];
    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=(float)total/n;
  printf("\n\nAverage Waiting Time=%f",avg_wt);
  printf("\nAverage Turnaround Time=%f\n",avg_tat);
}
```

OUTPUT:

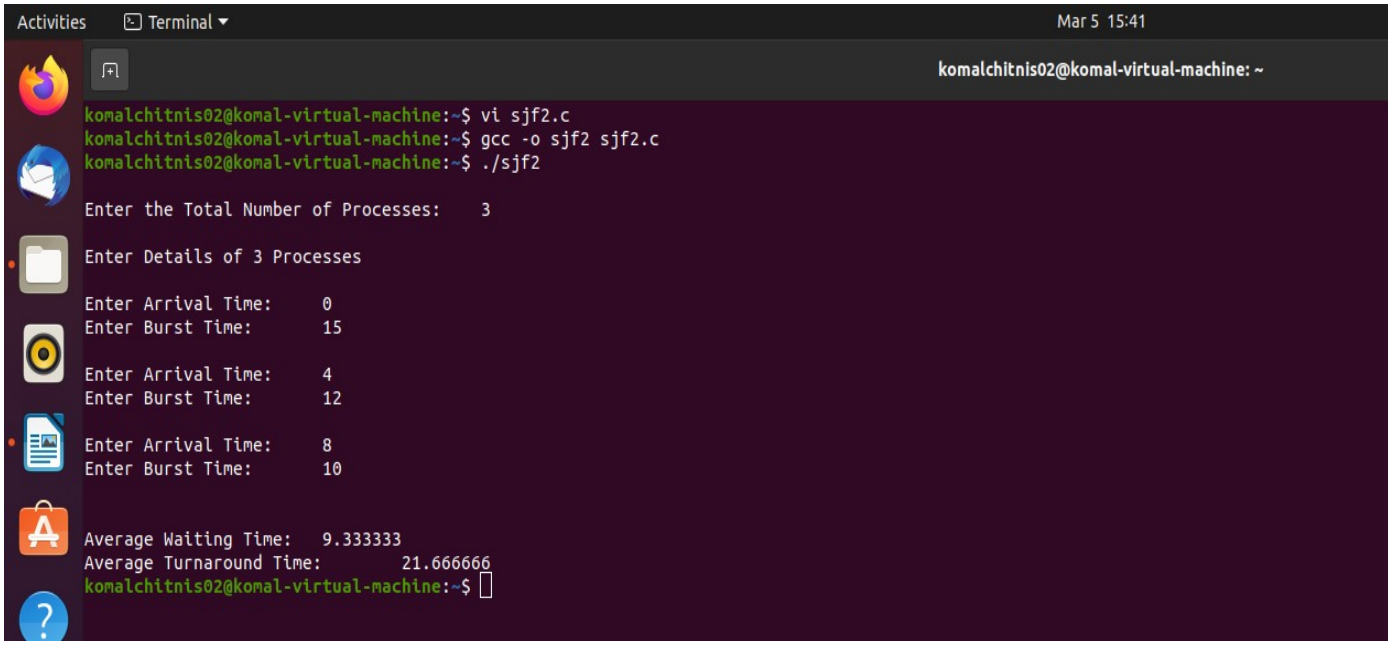


NON-PREMTIVE:

```
#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 Total Number of Processes:\t");
    scanf("%d", &limit);
    printf("\nEnter Details of %d Processes\n", limit);

    for(i = 0; i < limit; i++)
    {
        printf("\nEnter Arrival Time:\t");
        scanf("%d", &arrival_time[i]);
        printf("Enter Burst Time:\t");
        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:\t%lf\n", average_waiting_time);
    printf("Average Turnaround Time:\t%lf\n", average_turnaround_time);
    return 0;
}
```

OUTPUT:



ROUND ROBIN SCHEDULING IN C PROGRAMMING

```
#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 ID\t\tBurst Time\t Turnaround Time\t Waiting Time\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("\nProcess[%d]\t\t%d\t\t %d\t\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:\t%f", average_wait_time);
    printf("\nAvg Turnaround Time:\t%f\n", average_turnaround_time);
    return 0;
}
```

OUTPUT: