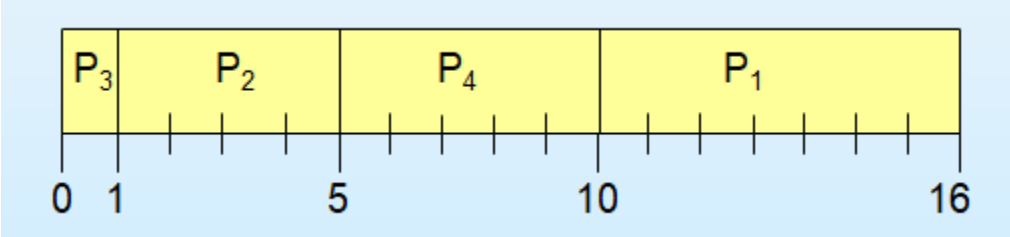
Experiment No: 02 (a)

Experiment Name: Implementation of Shortest Job First (SJF) Algorithm for Non-Primitive Approach Using C/C++ Where Processes Arrival Simultaneously.

SJF Non-Primitive Approach (Simultaneous Arrival) Algorithm Theoretical Explanation:

Process	Arrival Time	Burst Time
P1	0.0	6
P2	0.0	4
P3	0.0	1
P4	0.0	5

SJF (non-preemptive, simultaneous arrival)



Average waiting time = (0 + 1 + 5 + 10)/4 = 4

Average turn-around time = (1 + 5 + 10 + 16)/4 = 8

SJF (non-preemptive, simultaneous arrival) Algorithm Using C++ Code:

```
#include<bits/stdc++.h>
using namespace std;
int main ()
{
   int i,n,j,temp;
   int bt[100],wt[100],tat[100],p[100];
   float awt=0,atat=0;
   printf("Shortest Job First Scheduling Non-Primitive(Simultaneous Arrival)\n");
   printf("Enter the No. of processes :\n");
   scanf("%d",&n);
```

```
for(i=0; i<n; i++)
{
   //   printf("Enter the arrival time of %d process :\n",i+1);
   // scanf("%d",&at[i]);

   printf("Enter the burst time of %d process :\n",i+1);
   scanf("%d",&bt[i]);
   p[i]=i+1;
}
/*Sorting According to Burst Time*/
for(i=0; i<n; i++)
{
   for(j=i+1; j<n; j++)
   {
      if(bt[i]>bt[j])
      {
         temp=bt[j];
         bt[j]=bt[i];
         bt[i]=temp;
         temp=p[j];
         p[j]=p[i];
         p[i]=temp;
      }
   }
}
/*waiting time & turnaround time calculation of every process*/
wt[0]=0;
tat[0]=bt[0];
for(i=1; i<n; i++)
{
   wt[i]=bt[i-1]+wt[i-1];
```

```
        awt+=wt[i];
        tat[i]=tat[i-1]+bt[i];
        atat+=tat[i];
    }
    atat+=tat[0];
    atat/=n;
    awt/=n;
    printf("Process.\tB.T.\tW.T.\tT.A.T.\n");
    for(i=0; i<n; i++)
    {
        printf("P[%d]\t        %3d\t%3d\t%4d\n",p[i],bt[i],wt[i],tat[i]);
    }
    printf("Average Waiting Time: %0.3f\nAverage Turn Around Time:%0.3f",awt,atat);
    return 0;
}
```

Output:

```
Shortest Job First Scheduling Non-Primitive(Simultaneous Arrival)
Enter the No. of processes :
4
Enter the burst time of 1 process :
6
Enter the burst time of 2 process :
4
Enter the burst time of 3 process :
1
Enter the burst time of 4 process :
5
Process.        B.T.    W.T.    T.A.T.
P[3]               1       0        1
P[2]               4       1        5
P[4]               5       5       10
P[1]               6      10       16
Average Waiting Time: 4.000
Average Turn Around Time:8.000
Process returned 0 (0x0)   execution time : 9.839 s
Press any key to continue.
```

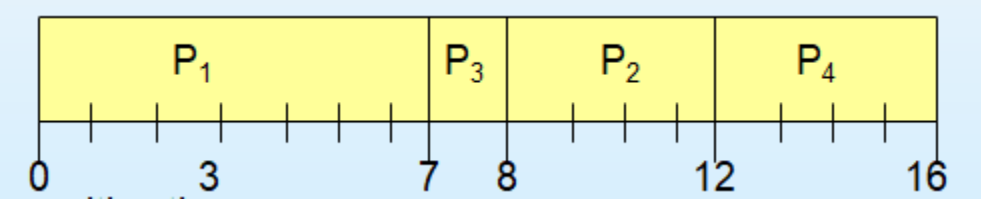
Experiment No: 02 (b)

Experiment Name: Implementation of Shortest Job First (SJF) Algorithm for Non-Primitive Approach Using C/C++ When Arrival is Different.

SJF Non-Primitive Approach (Different Arrival) Algorithm Theoretical Explanation:

Process	Arrival Time	Burst Time
P1	0	7
P2	2	4
P3	4	1
P4	5	4

SJF (non-preemptive, varied arrival times)



Average waiting time

= ((0 – 0) + (8 – 2) + (7 – 4) + (12 – 5))/4
= (0 + 6 + 3 + 7)/4 = 4

Average turn-around time:

= ((7 – 0) + (12 – 2) + (8 - 4) + (16 – 5))/4
= (7 + 10 + 4 + 11)/4 = 8

SJF (non-preemptive, varried arrival) Algorithm Using C++ Code:

```
#include<bits/stdc++.h>
using namespace std;
int main ()
{
   int i,n,j,temp,ta=0,min;
   int bt[100],wt[100],tat[100],p[100],at[100],sum=0,btime=0,k=1;
   float awt=0,atat=0;
   printf("Shortest Job First Scheduling Non-Primitive(Varried Arrival)\n");
   printf("Enter the No. of processes :\n");
   scanf("%d",&n);
   for(i=0; i<n; i++)
   {
      printf("Enter the arrival time of %d process :\n",i+1);
      scanf("%d",&at[i]);
      printf("Enter the burst time of %d process :\n",i+1);
      scanf("%d",&bt[i]);
      p[i]=i+1;
   }
   /*Sorting According to Burst Time because arrival time is already sorted*/
   for(j=0; j<n; j++)
   {
      btime=btime+bt[j];
      min=bt[k];
      for(i=k; i<n; i++)
```

```
    {
        if (at[i]<=btime && bt[i]<min)
        {
            temp=p[k];
            p[k]=p[i];
            p[i]=temp;
            temp=at[k];
            at[k]=at[i];
            at[i]=temp;
            temp=bt[k];
            bt[k]=bt[i];
            bt[i]=temp;
        }     }
    k++; }
wt[0]=0;
for(i=1; i<n; i++)
{
    sum=sum+bt[i-1];
    wt[i]=sum-at[i];
    awt=awt+wt[i];
}
for(i=0; i<n; i++)
{
    ta=ta+bt[i];
    tat[i]=ta-at[i];
```

```
    atat=atat+tat[i];
  }
  atat/=n;
  awt/=n;
  printf("Process.\tB.T.\tW.T.\tT.A.T.\n");
  for(i=0; i<n; i++)
  {
    printf("P[%d]\t      %3d\t%3d\t%4d\n",p[i],bt[i],wt[i],tat[i]);
  }
  printf("Average Waiting Time: %0.3f\nAverage Turn Around Time:%0.3f",awt,atat);
  return 0;
}
```

Output:

```
Shortest Job First Scheduling Non-Primitive(Varried Arrival)
Enter the No. of processes :
4
Enter the arrival time of 1 process :
0
Enter the burst time of 1 process :
7
Enter the arrival time of 2 process :
2
Enter the burst time of 2 process :
4
Enter the arrival time of 3 process :
4
Enter the burst time of 3 process :
1
Enter the arrival time of 4 process :
5
Enter the burst time of 4 process :
4
Process.        B.T.    W.T.    T.A.T.
P[1]               7       0        7
P[3]               1       3        4
P[2]               4       6       10
P[4]               4       7       11
Average Waiting Time: 4.000
Average Turn Around Time:8.000
Process returned 0 (0x0)   execution time : 10.868 s
Press any key to continue.
```