LAB - 3

PAGE NO :
DATE :

① Write a C program to simulate the following non-preemptive CPU scheduling algorithm to find turnaroundtime and waiting time.

1) FCFS.

```
#include <stdio.h>

int main()
{
    int p[10], at[10], bt[10], ct[10], tat[10],
    wt[10], i, j, temp = 0, n;
    float awt = 0, atat = 0;
    printf("enter no.of process:");
    scanf("%d", &n);
    printf("enter %d process:", n);
    for(i=0; i<n; i++)
   {
        scanf("%d", &p[i]);
   }
   printf("enter %d arrivaltime:", n);
   for(i=0; i<n; i++)
   {
        scanf("%d", &at[i]);
   }
   printf("enter %d burst time:", n);
   for(i=0; i<n; i++)
   {
        scanf("%d", &bt[i]);
   }
   ct[0]= at[0]+bt[0];
   for(i=1; i<n; i++)
  {
        temp = 0;
```

PAGE NO :
DATE :

```
if (ct[i-1] < at[i])
{
        temp = at[i] - ct[i-1];
}
    ct[i] = ct[i-1] + bt[i] + temp;
}
printf("\np\t  A.T\t  B.T\t C.T\t
            TAT\t    WT");
for (i=0; i<n; i++)
{
    tat[i] = ct[i] - at[i];
    wt[i] = tat[i] - bt[i];
    atat + = tat[i];
    awt + = wt[i];
}
atat = atat/n;
awt = awt/n;
for(i=0; i<n; i++)
{
    printf("\nP %d\t %d\t %d\t %d\t
            %d\t  %d", p[i], at[i], bt[i],
            ct[i], tat[i], wt[i]);
}
printf("\n average turnaround time is %f",
        atat);
printf("\n average  waiting  time is %f",
        awt);
return 0;
}
```

Output :-

```
enter  no. of   processes : 4
enter  4  process : 1 2 3 4
enter  4  arrival time : 0 1 5 6
enter  4  burst time : 2 2 3 4

P      A.T    B.T   C.T   TAT   WT
P1     0      2     2     2     0
P2     1      2     4     3     1
P3     5      3     8     3     0
P4     6      4     12    6     2
average turnaround time :: 3.500000
average waiting time :: 0.750000
```

② SJF (Non preemptive):

```
#include    <stdio.h>
#include    <conio.h>
#include    <stdlib.h>
void  swap (int  *x, int  *y)
{
        int   temp = *x;
              *x = *y;
              *y = temp;
}
void  Sortat (int  p[], int at[], int  bt[], i
{
    int   i, j;
    for(i=0;  i<n; i++)
    {
        for(j=i+1; j<n; j++)
        {
            if (at[i] > at[j])
            {
                  swap (&p[i], &p[j]);
                 swap (&at[i], &at[j]);
                 swap (&bt[i], &bt[j]);
            }
            else  if (at[i] = = at[j])
            {
                  if (bt[i] > bt[j])
                  {
                          swap (&p[i], &p[j]);
                         swap (&at[i], &at[j]);
                        swap (&bt[i], &bt[j]);
                  }
    }   }   }
```

```
void tatwt(int ct[], int at[], int bt[], int tat[],
           int wt[], int n)
{
        int i;
        for(i=0; i<n; i++)
        {
            tat[i] = ct[i] - at[i];
            wt[i] = tat[i] - bt[i];
        }
}
int main()
{
    int *p, *at, *bt, *tat, *wt, *ct, pos, i, j,
    min = 1000, n;
    float awt = 0, atat = 0;
    printf("\n enter the number of process:");
    scanf("%d", &n);
    p = (int*) malloc (n*sizeof(int));
    at = (int*) malloc (n*sizeof(int));
    bt = (int*) malloc (n*sizeof(int));
    ct = (int*) malloc (n*sizeof(int));
    wt = (int*) malloc (n*sizeof(int));
    tat = (int*) malloc(n*sizeof(int));
    printf("enter the process\n");
    for(i=0; i<n; i++)
    {
        scanf("%d", &p[i]);
    }
    printf("enter the arrival time\n");
    for(i=0; i<n; i++)
    {
        scanf("%d", &p[i]);
    }
```

```
printf("enter the arrival time\n");
for(i=0;i<n;i++)
{
    scanf("%d",&at[i]);
}
printf("enter the burst time\n");
for(i=0;i<n;i++)
{
    scanf("%d",&bt[i]);
}
Sortat(p,at,bt,n);
ct[0]=at[0]+bt[0];
for(i=1;i<n;i++)
{
    for(j=1;j<n;j++)
    {
        if(at[j]<=ct[i-1])
        {
            if(bt[j]<min)
            {
                min=bt[j];
                pos=j;
            }
        }
    }
    Swap(&p[i],&p[pos]);
    Swap(&at[i],&at[pos]);
    Swap(&bt[i],&bt[pos]);
    min=1000;
    ct[i]=ct[i-1]+bt[i];
}
tatwt(ct,at,bt,tat,wt,n);
printf("\np\t at\t bt\t ct\t tat\t wt\n");
for(i=0;i<n;i++)
```

```
}
    printf("\n%d\t %d\t %d\t %d\t %d\t
              %d", p[i], at[i], bt[i], ct[i], tat[i], wt[i]);
}
for(i=0; i<n; i++)
{
      atat += tat[i];
      awt += wt[i];
}
 atat = atat/n;
 awt = awt/n;
 printf("\n    avg    tat = %.2f and avg wt=%.2f",
        atat, awt);
 return 0;
}
```

Output:

```
enter the number of process: 4
enter the process
1  2 3 4
enter the arrival time
0   1  4 6
enter the burst time
3  6   4  2
p      at     bt       ct     tat     wt
1      0      3        3      3       0
2      1      6        9      8       2
4      6      2        11     5       3
3      4      4        15     11      7
 avg  tat = 6.75          avgwt = 3.00
```

[illegible]
16/5/2024