

1. Write a C program to simulate the following non-pre-emptive CPU scheduling algorithm to find turnaround time and waiting time.
FCFS
SJF (pre-emptive & Non-pre-emptive)

Code:

```
#include <stdio.h>
int at[10], pt[10], ia, ip, n;
int tat[10], wt[10], it, iw, pos, j, i;
float atat = 0, awt = 0;
void fcfs()
{
    int t;
    printf("Enter number of processes: ");
    scanf("%d", &n);

    printf("Enter arrival times:\n");
    for (ia = 0; ia < n; ia++)
        scanf("%d", &at[ia]);

    printf("Enter process times:\n");
    for (ip = 0; ip < n; ip++)
        scanf("%d", &pt[ip]);

    if (at[0] == at[1])
    {
        t = pt[1];
        pt[1] = pt[0];
        pt[0] = t;
    }

    if (at[0] != 0)
        tat[0] = at[0];

    for (it = 0; it < n; it++)
        tat[it] = 0;

    int i = 0;
    for (it = 0; it < n; it++)
    {
        while (i <= it)
            tat[it] += pt[i++];
        i = 0;
    }
}
```

```

for (it = 0; it < n; it++)
    tat[it] = tat[it] - at[it];

for (ia = 0; ia < n; ia++)
    wt[ia] = tat[ia] - pt[ia];

for (i = 0; i < n; i++)
{
    atat += tat[i];
    awt += wt[i];
}

atat = atat / n;
awt = awt / n;

for (i = 0; i < n; i++)
{
    printf("P%d\t%d\t%d\n", i, tat[i], wt[i]);
}

printf("Average TAT=%.2f\nAverage WT=%.2f\n", atat, awt);
}

```

```

void srff()
{
    int rt[10], endTime, i, smallest;
    int remain = 0, time, sum_wait = 0, sum_turnaround = 0;
    printf("Enter no of Processes : ");
    scanf("%d", &n);
    printf("Enter arrival times\n");
    for (i = 0; i < n; i++)
    {
        scanf("%d", &at[i]);
    }
    printf("Enter Process times \n");
    for (i = 0; i < n; i++)
    {
        scanf("%d", &pt[i]);
        rt[i] = pt[i];
    }
    rt[9] = 9999;
    for (time = 0; remain != n; time++)
    {
        smallest = 9;
        for (i = 0; i < n; i++)
        {

```

```

        if (at[i] <= time && rt[i] < rt[smallest] && rt[i] > 0)
        {
            smallest = i;
        }
    }
    rt[smallest]--;
    if (rt[smallest] == 0)
    {
        remain++;
        endTime = time + 1;
        printf("\nP%d %d %d", smallest + 1, endTime - at[smallest], endTime - pt[smallest] -
at[smallest]);
        sum_wait += endTime - pt[smallest] - at[smallest];
        sum_turnaround += endTime - at[smallest];
    }
}
printf("\n\nAverage waiting time = %f\n", sum_wait * 1.0 / n);
printf("Average Turnaround time = %f", sum_turnaround * 1.0 / n);
}

```

```

void sjf()
{
    int completed = 0;
    int currentTime = 0;
    int complete[n], ct[n];

    printf("Enter number of processes: ");
    scanf("%d", &n);

    printf("Enter arrival times:\n");
    for (int ia = 0; ia < n; ia++)
        scanf("%d", &at[ia]);

    printf("Enter process times:\n");
    for (int ip = 0; ip < n; ip++)
        scanf("%d", &pt[ip]);

    for (int i = 0; i < n; i++)
    {
        complete[i] = 0;
        ct[i] = 0;
    }

    while (completed != n)
    {
        int shortest = -1;

```

```

int min_bt = 9999;

for (int i = 0; i < n; i++)
{
    if (at[i] <= currentTime && complete[i] == 0)
    {
        if (pt[i] < min_bt)
        {
            min_bt = pt[i];
            shortest = i;
        }
        if (pt[i] == min_bt)
        {
            if (at[i] < at[shortest])
            {
                shortest = i;
            }
        }
    }
}

if (shortest == -1)
{
    currentTime++;
}
else
{
    ct[shortest] = currentTime + pt[shortest];
    tat[shortest] = ct[shortest] - at[shortest];
    wt[shortest] = tat[shortest] - pt[shortest];
    complete[shortest] = 1;
    completed++;
    currentTime = ct[shortest];
}
}

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

atat = atat / n;
awt = awt / n;

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

```

```

    {
        printf("P%d\t%d\t%d\n", i, tat[i], wt[i]);
    }

    printf("\nAverage TAT = %f\nAverage WT = %f\n", atat, awt);
}

void main()
{
    int op = 1, x;
    printf("1.FCFS \n2.SJF \n3.SRTF\n");
    scanf("%d", &x);
    switch (x)
    {
        case 1:
            fcfs();
            break;
        case 2:
            sjf();
            break;

        case 3:
            srtf();
            break;

        default:
            printf("Invalid option \n");
    }
}

```

21-08-2005

Write a C programming to simulate CPU scheduling

① void FFS()

```

{
    int i=0;
    for (it=0; it<n; it++)
        tat[it]=0;

    for (it=0; it<n; it++)
    {
        while (ic == it)
        {
            tat[it] += pt[it];
            i++;
        }
    }

```

```

    for (it=0; it<n; it++)
        tat[it] = tat[it] - at[it];

```

```

    for (ia=0; ia<n; ia++)
        wt[ia] = tat[ia] - pt[ia];

```

```

    for (i=0; i<n; i++)
    {
        atat += tat[i];
        awt += wt[i];
    }

```

```

    atat = atat/n;
    awt = awt/n;

```

```

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

```

```

    {
        printf("pid %d %d %d\n", i, tat[i], wt[i]);
    }

```

```

    printf("Average TAT = %d %d\n", atat, awt);
}

```

2.

void SJF()

```

{
    int completed=0;

```

```

    int currentline=0;

```

```

    int complete[n], ct[n];

```

```

    printf("Enter number of processes: ");

```

```

    scanf("%d", &n);

```

```

    printf("Enter arrival times: ");

```

```

    scanf("%d", &at);

```

```

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

```

```

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

```

```

    }
    printf("Enter process times: ");

```

```

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

```

```

    {
        scanf("%d", &pt[i]);
    }
}

```



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

```
{
  complete[i]=0;

```

```
  ct[i]=0;

```

```
}
```

```
while (completed != n)
```

```
{
```

```
  int shortest=-1;

```

```
  int min_bt=9999;

```

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

```
  {
    if (at[i] <= currentTime && complete[i]==0)

```

```
    {
      if (pt[i] < min_bt)

```

```
      {
        min_bt = pt[i];

```

```
        shortest = i;

```

```
      }
      if (pt[i] == min_bt)

```

```
      {
        if (at[i] < at[shortest])

```

```
        {
          shortest = i;

```

```
        }

```

```
      }
    }

```

```
    if (shortest == -1)

```

```
    {
      currentTime++;

```

```
    }

```

```
    else

```

```
    {
      ct[shortest] = currentTime + pt[shortest];

```

```
      fat[shortest] = ct[shortest] - at[shortest];

```

```
      wt[shortest] = fat[shortest] - pt[shortest];

```

```
      complete[shortest] = 1;

```

```
      completed++;

```

```
      currentTime = ct[shortest];

```

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

```

```
  {
    atot += fat[i];

```

```
    wot += wt[i];

```

```
  }

```

```
  atot = atot / n;

```

```
  wot = wot / n;

```

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

```

```
  {
    printf(" %d %d %d %d %d\n", i, fat[i], wt[i]);

```



```
printf("\n Average TAT = %0.2f \n Average WT = %0.2f", at/n, wt/n);
```

```
void sort()
```

```
{
    int at[10], endTime, i, smallest;
    int remain=0, time, sum_wait=0, sum_turnaround=0;
    printf("Enter number of processes: ");
    scanf("%d", &n);
    printf("Enter arrival time in ");
    for(i=0; i<n; i++)
        scanf("%d", &at[i]);
    printf("Enter processes time in ");
    for(i=0; i<n; i++)
        scanf("%d", &pt[i]);
    at[9] = 9999;
    for(time=0; remain!=n; time++)
    {
        smallest=9;
        for(i=0; i<n; i++)
        {
            if(at[i] <= time && at[i] < at[smallest] && at[i] > 0)
            {
                smallest=i;
            }
            at[smallest]--;
            if(at[smallest] == 0)
            {
                remain++;
                endTime = time + 1;
                printf("\n P %d %d %d", smallest, endTime,
                    at[smallest], endTime - pt[smallest] - at[smallest]);
                sum_wait += endTime - pt[smallest] - at[smallest];
                sum_turnaround += endTime - at[smallest];
            }
        }
    }
}
```

```
printf("\n Average waiting time = %0.2f \n", sum_wait/n);
printf("\n Average TAT = %0.2f \n", sum_turnaround/n);
```

1. FCFS
2. SJF
3. SRTF

1. Enter n

Enter
Enter

P1

P2

P3

A TAT
A WT

Ent

P1

P2

P3

A TAT

Ent

Enter

P1

P2

P3

A T

A

1. FCFS
2. SJF
3. SRTF

1. Enter no. of processes 3.

Enter Arrival times 0 0 1
Enter process times 8 4 1

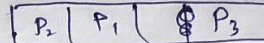
P1 12 4

P2 4 0

P3 12 11

ATAT = 9.3.

AWT = 5.



2.

Enter no. of processes 3.

Enter Arrival times 0 0 1
Enter process times 8 4 1

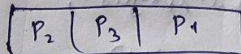
P1 13 5

P2 4 0

P3 4 3

ATAT = 7

AWT = 2.67.



3.

Enter no. of processes 3.

Enter Arrival times 0 0 1
Enter process times 8 4 1

P1 13 5

P2 4 0

P3 4 3

ATAT = 7

AWT = 2.67.

