16/5/24

PAGE NO.:
DATE

1) Write a c program to stimulate the following non pre-emptive CPU scheduling algorithm to find turnaround time and waiting time.

a) FCFS:

```
#include <stdio.h>
int main()
{
    int p[10], at[10], bt[10], ct[10], tat[10], wt[10];
    int i, j, temp=0, n;
    float awt=0, atat=0;
    printf("Enter nor of processes \n");
    scanf("%d", &n);
    printf("Enter %d process: ", n);
    for(i=0; i<n; i++){
        scanf("%d", &p[i]);
    }
    printf("Enter %d arrival time: ", 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;
        if(ct[i-1]<at[i]){
            temp=at[i]-ct[i-1];
        }
        ct[i]=ct[i-1]+bt[i]+temp;
    }
    printf("\n p \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("\n P %d \t %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 nor of process: 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	AT	BT	CT	TAT	WT
P_1	0	2	2	2	0
P_2	1	2	4	3	1
P_3	5	3	8	3	0
P_4	6	4	12	6	2

Average turnaround time is 3.500000
Average waiting time is 0.750000

b) SJF (Non-preemptive)

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
void swap(int *x, int *y){
    int temp = *x;
    *x = *y;
    *y = temp;
}
void sortat(int p[], int at[], int bt[], int n)
{
    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 nor 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 process");
    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 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("\n p\t at\t bt\t ct\t tat\t wt");
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 & avg wt = %.2f", atat,
    awt);
return 0;
}
```

→ Output:

Enter the number of process: 5

Enter process: 1 2 3 4 5

Enter arrival time: 2 1 4 0 2

Enter burst time: 1 5 1 6 3

p	at	bt	ct	tat	wt
4	0	6	6	6	0
5	2	1	7	5	4
3	4	1	8	4	3
1	2	3	11	9	6
2	1	5	16	15	10

Avg tat = 7.80 and
Avg wt = 4.60

16/5/2024