MPS :

```
#include <stdio.h>

struct process {
    int at, st, ft, status, cpu;
} ready_list[10];

int n, h;

int dispatcher(int time) {
    for (int i = 0; i < n; i++) {
        if (ready_list[i].status == 0 && ready_list[i].at <= time) {
            return i;
        }
    }
    return -1;
}

int main() {
    printf("Enter number of processes and CPUs: ");
    scanf("%d %d", &n, &h);

    for (int i = 0; i < n; i++) {
        printf("Process %d: Enter Arrival Time and Service Time: ", i + 1);
        scanf("%d %d", &ready_list[i].at, &ready_list[i].st);
        ready_list[i].status = 0;
    }

    int cur_time[10] = {0}; // Initialize current times for each CPU
    int completed = 0;

    while (completed < n) {
        for (int j = 0; j < h && completed < n; j++) {
            int pid = dispatcher(cur_time[j]);
            if (pid != -1) {
                ready_list[pid].ft = cur_time[j] + ready_list[pid].st;
                ready_list[pid].cpu = j + 1;
                ready_list[pid].status = 1;
                cur_time[j] += ready_list[pid].st;
                completed++;
            }
        }
    }

    printf("Process\tAT\tST\tFT\tCPU\tTT\tWT\n");
    for (int i = 0; i < n; i++) {
        int tt = ready_list[i].ft - ready_list[i].at;
        int wt = tt - ready_list[i].st;
```

```
        printf("%d\t%d\t%d\t%d\t%d\t%d\t%d\n", i + 1, ready_list[i].at, ready_list[i].st,
ready_list[i].ft, ready_list[i].cpu, tt, wt);
    }

    return 0;
}
```