15/6/23
2) write a c program to execute fcfs.
SRTF for process scheduling.

```
#include<stdio.h>
int at [20], cput[20];
 void main(){
int at [50], cput[50], tat[50], wt [50], n, current time=0
 void Shortest_job();
     int Completed=0;
     int remaining[n];
     int Completed_t[n];
for (int i=0;i<n;i++),
         remaining[i]=cput[i];}
 while(Completed !=n){.
     int Shortest=-1;
   int min_time =1000;
 for (int i=0; i<n;i++)
if (at[i]<=current_time && output [i]<min_time
        && remaining [i]>0){
             Shortest =i;
        min_time = cput[i]; }}
     if (Shortest ==-1){
          current_time++;
          Continue ;}
Completed_t[Shortest]=current_time + remaining [Shortest]
  current_time += remaining [Shortest];
```

```
remaining[shortest] =0;
tat [Shortest] = completed - t [shortest] - at
        [Shortest];
completed++; }}
int main(){
    Printf ("Enter no of processes");
    scanf ("%d", &n);
    Printf ("Enter arrival & cpu time");
  for (int i=0; i<n; i++){
   scanf ("%d %d", &at [i], &cpu_ti[i] }
 }
 Shortest_job ();
    float count =0; count w=0;
  for [int =0; i<n; i++){
        count_t = tat [i];
        countat ~~ct~~ = wt[i];
Print f("p %d: waiting = %d, turnaround = %d:
        i, wt[i], tat [i]);
Printf ("Avg wt = %f", (float) (count w)/n);
   Printf ("\n Avg TAT = %f (float ) (count)/n);
   return 0;
 }
```

FCFS

```
#include <Stdio.h>
#include <Stdlib.h>
int main (){
   int n;
  Printf ("Enter no of processes \n");
  Scanf ("%d", &n)
  int * tat arr = (int *) malloc (n * size of (int));
 int * Wt arr = (int *) malloc (n * size of (int));
 int *at arr = (int *) malloc (n * size of (int));
 int * cpu-atarr = (int *) malloc (n * size of (int));
 Printf ("Enter arrival & burst \n");
    int awt = 0;
    int at at = 0;
    int Sumtime = 0;
 for (int i=0; i<n; i++){
 Scanf ("%d %d", at arr +i, cput-arr +i); }
for (i=0; i<n-1; i++){
      for (int j = i+1; j<n; j++) }
      if (* (atarr + j) < * (atarr + i) ){
      int temp = * (atarr + i);
        * (at arr + i) = * (at arr + j);
         * (at arr + j) = temp;
           temp = * (cput arr + i);
```

```
* (cput arr + i) = * [cput arr + j],
* (cput arr +j) = temp ; }}
for (int i=0; i<n , i++)
   Sumtime = * [cput arr + i),
    * (tatarr + i ) = Sum time,
    * (wt arr + i) = * (tat arr + i) - *
                          (cbwt arr + i);
  awt + = * (wt arr + i);
  wttat + = * (tat arr + i); }
for (int i = 0; i<n ; i++) {
 Print f ("p %d, waiting = %d (tatarond = %d)
      i , * (wt arr + i), * tat arr + i)),
 }
Printf (" the average wt = %f \n the average tt
      = %f \n", (float awt (n , (float) awtat (n,
  (float ) Sumtime / n ;
      return 0,
```

1 SRETF -

```
int at [50], cpbt [50], tat [50], wt [50], n,
        current-time =0;
void shortest job_t () }
        int completed =0;
        int remaining [n];
        int completed-t [n];
    for (int i=0; i<n; i++ ){
        remaining [i] = cpbt [i]; }
      while (completed != n) {
            int shortest -1;
            int min-time =1000;
        for (int i=0; i<n; i++) }
        if (at[i] <= current-time &&
    remaining [i] < min-time && remaining [i]>0) {
          shortest = i;
        min-time = remaining [i];
          }
      }
      if (shortest == -1) {
              current -time ++;
               Continue
              }
```

```
currant-time t = 1;
remaining [shortest] - = 1;
        if (remainis [shortest] = = 0) {
            completed ++;
      completed - t [shortest] = currant - time;
   tat [shortest] = completed - t (shortest) - at (shortest);
  wt [shortest] = tat [shortest] - Cpbut (shortest);
    }
  }
```

[illegible]