//round robin---

```
#include<stdio.h>


int main()
{
   int n;
   scanf("%d", &n);
   int process[n], burstTime[n];
   for(int i = 0 ; i < n ; i++)
   {
      scanf("%d %d", &process[i], &burstTime[i]);
   }
   int timeQuantum;
   scanf("%d", &timeQuantum);

   //Sort the array in ascending order
   int temp;
   for (int i = 0 ; i < n ; i++)
   {
      for (int j = i+1 ; j < n ; j++)
      {
         if(process[i] > process[j])
         {
            temp = process[i];
            process[i] = process[j];
            process[j] = temp;

            temp = burstTime[i];
            burstTime[i] = burstTime[j];
            burstTime[j] = temp;
```

```
      }
    }
}

int temporaryArray[n];
for(int i = 0 ; i < n ; i++)
{
   temporaryArray[i] = burstTime[i];
}

int completionTime[n], timestamp = 0;
while(true)
{
   for(int i = 0 ; i < n ; i++)
   {
      if(temporaryArray[i] > timeQuantum)
      {
         temporaryArray[i] =temporaryArray[i]-timeQuantum;
         timestamp = timestamp + timeQuantum;
      }
      else
      {
         if(temporaryArray[i] != 0)
         {
            timestamp = timestamp + temporaryArray[i];
            temporaryArray[i] = 0;
            completionTime[i] = timestamp;
         }
      }
   }
```

```
        bool flag = false;
        for(int i = 0 ; i < n ; i++)
        {
            if(temporaryArray[i] != 0)
                flag = true;
        }
        if(flag == false)
            break;
    }


    int turnAroundTime[n], waitingTime[n];


    for(int i = 0 ; i < n ; i++)
    {
        turnAroundTime[i] = completionTime[i] - 0;  // Assumption: Arrival Time = 0 for all the processes
        waitingTime[i] = turnAroundTime[i] - burstTime[i];
    }


    printf("PID\tBT\tCT\tTAT\tWT\n");
    for (int i = 0; i < n; i++) {
        printf("%d\t%d\t%d\t%d\t%d\n", process[i], burstTime[i], completionTime[i],
turnAroundTime[i], waitingTime[i]);
    }


    float averageTAT, averageWT;
    float sumTAT = 0, sumWT = 0;
    for(int i = 0 ; i < n ; i++)
    {
        sumTAT = sumTAT + turnAroundTime[i];
        sumWT = sumWT + waitingTime[i];
```

```
    }

    // printf("Sum of Turn Around Time = %f \n", sumTAT);
    // printf("Sum of Waiting Time = %f \n", sumWT);

    averageTAT = sumTAT / n;
    averageWT = sumWT / n;

    printf("Average Turn Around Time = %f \n", averageTAT);
    printf("Average Waiting Time = %f \n", averageWT);
    return 0;
}
```

//sjf.....

```
#include<stdio.h>
int main()
{
    int n,i,j;
    scanf("%d", &n);
    int process[n], burstTime[n];
    for(i = 0 ; i < n ; i++)
    {
        scanf("%d %d", &process[i], &burstTime[i]);
    }
```

```
//Sort the array in ascending order
int temp;
for (i = 0 ; i < n ; i++)
{
   for (j = i+1 ; j < n ; j++)
   {
      if(burstTime[i] > burstTime[j])
      {
         temp = burstTime[i];
         burstTime[i] = burstTime[j];
         burstTime[j] = temp;

         temp = process[i];
         process[i] = process[j];
         process[j] = temp;
      }
   }
}

int completionTime[n];

completionTime[0] = burstTime[0];
int timestamp = completionTime[0];

for(i = 1 ; i < n ; i++)
{
   timestamp = timestamp + burstTime[i];
   completionTime[i] = timestamp;
}

//Sort the array in ascending order
```

```
for (i = 0 ; i < n ; i++)
{
   for (j = i+1 ; j < n ; j++)
   {
      if(process[i] > process[j])
      {
         temp = process[i];
         process[i] = process[j];
         process[j] = temp;

         temp = burstTime[i];
         burstTime[i] = burstTime[j];
         burstTime[j] = temp;

         temp = completionTime[i];
         completionTime[i] = completionTime[j];
         completionTime[j] = temp;
      }
   }
}

int turnAroundTime[n], waitingTime[n];

for(i = 0 ; i < n ; i++)
{
   turnAroundTime[i] = completionTime[i] - 0;  // Assumption: Arrival Time = 0 for all the processes
   waitingTime[i] = turnAroundTime[i] - burstTime[i];
}

printf("PID\tBT\tCT\tTAT\tWT\n");
for (i = 0; i < n; i++) {
```

```
        printf("%d\t%d\t%d\t%d\t%d\n", process[i], burstTime[i], completionTime[i],
turnAroundTime[i], waitingTime[i]);
    }

    float averageTAT, averageWT;
    float sumTAT = 0, sumWT = 0;
    for(i = 0 ; i < n ; i++)
    {
        sumTAT = sumTAT + turnAroundTime[i];
        sumWT = sumWT + waitingTime[i];
    }

    // printf("Sum of Turn Around Time = %f \n", sumTAT);
    // printf("Sum of Waiting Time = %f \n", sumWT);

    averageTAT = sumTAT / n;
    averageWT = sumWT / n;

    printf("Average Turn Around Time = %f \n", averageTAT);
    printf("Average Waiting Time = %f \n", averageWT);
    return 0;
}
```

//fcfs code:

```
#include<stdio.h>
int main()
{
   int n,j,i;
   scanf("%d", &n);
   int process[n], arrivalTime[n], burstTime[n];
   printf("process : at : bt:");
   for(i = 0 ; i < n ; i++)
   {
      scanf("%d %d %d", &process[i], &arrivalTime[i], &burstTime[i]);
   }

   //Sort the array in ascending order
   int temp;
   for ( i = 0 ; i < n ; i++)
   {
      for (j = i+1 ; j < n ; j++)
      {
         if(arrivalTime[i] > arrivalTime[j])
         {
            temp = arrivalTime[i];
            arrivalTime[i] = arrivalTime[j];
            arrivalTime[j] = temp;

            temp = burstTime[i];
            burstTime[i] = burstTime[j];
            burstTime[j] = temp;
```

```
            temp = process[i];
            process[i] = process[j];
            process[j] = temp;
        }
    }
}

int completionTime[n];

int timestamp = arrivalTime[0];
timestamp = timestamp + burstTime[0];
completionTime[0] = timestamp;

for(i = 1 ; i < n ; i++)
{
    if(arrivalTime[i] > timestamp)
        timestamp = arrivalTime[i];
    timestamp = timestamp + burstTime[i];
    completionTime[i] = timestamp;
}

//Sort the array in ascending order
for (i = 0 ; i < n ; i++)
{
    for ( j = i+1 ; j < n ; j++)
    {
      if(process[i] > process[j])
      {
            temp = process[i];
            process[i] = process[j];
            process[j] = temp;
```

```
            temp = arrivalTime[i];
            arrivalTime[i] = arrivalTime[j];
            arrivalTime[j] = temp;

            temp = burstTime[i];
            burstTime[i] = burstTime[j];
            burstTime[j] = temp;

            temp = completionTime[i];
            completionTime[i] = completionTime[j];
            completionTime[j] = temp;
        }
      }
    }

    int turnAroundTime[n], waitingTime[n];

    for(i = 0 ; i < n ; i++)
    {
       turnAroundTime[i] = completionTime[i] - arrivalTime[i];
       waitingTime[i] = turnAroundTime[i] - burstTime[i];
    }

    printf("PID\tAT\tBT\tCT\tTAT\tWT\n");
    for (i = 0; i < n; i++) {
       printf("%d\t%d\t%d\t%d\t%d\t%d\n", process[i], arrivalTime[i], burstTime[i],
completionTime[i], turnAroundTime[i], waitingTime[i]);
    }

    float averageTAT, averageWT;
```

```
    float sumTAT = 0, sumWT = 0;
    for(i = 0 ; i < n ; i++)
    {
        sumTAT = sumTAT + turnAroundTime[i];
        sumWT = sumWT + waitingTime[i];
    }

    // printf("Sum of Turn Around Time = %f \n", sumTAT);
    // printf("Sum of Waiting Time = %f \n", sumWT);

    averageTAT = sumTAT / n;
    averageWT = sumWT / n;

    printf("Average Turn Around Time = %f \n", averageTAT);
    printf("Average Waiting Time = %f \n", averageWT);
    return 0;
}
```