Vẽ lưu đồ RR:

A diagram of a flowchart

Description automatically generated

Chạy test case với 5 tiến trình như hình

RR(quantum time = 4)

A table with numbers and text

Description automatically generated

Kết quả giản đồ gantt:

A number of numbers and letters

Description automatically generated with medium confidence

Thời gian chờ:

P1 = 4 + 10 = 14, P2 = 2 + 8 = 10, P3 = 7 + 14 = 21, P4 = 10 , P5 = 14 + 4 = 18

* Thời gian chờ trung bình: (14 + 10 + 21 + 10 + 18)/5 = 14,6

Thời gian hoàn thành:

P1 = 26, P2 = 19 -2 = 17, P3 = 34 - 5 = 29, P4 = 22 – 9 = 13 , P5 = 36 - 12 = 24

* Thời gian chờ trung bình: (26 + 17 + 29 + 13 + 24)/5 = 21,8

A screen shot of a computer

Description automatically generated

RR (quantum time = 10)

|  |  |  |
| --- | --- | --- |
| **Process** | **Arrival Time** | **Burst Time** |
| P1 | 0 | 10 |
| P2 | 2 | 29 |
| P3 | 4 | 3 |
| P4 | 5 | 7 |
| P5 | 7 | 12 |

Kết quả giản đồ gantt:

Ảnh có chứa văn bản, Phông chữ, số, ảnh chụp màn hình

Mô tả được tạo tự động

Thời gian chờ:

P1 = 0, P2 = 8 + 20 + 2 = 30, P3 = 16, P4 = 18 , P5 = 23 + 10 = 33

* Thời gian chờ trung bình: (0 + 30 + 16 + 18 + 33)/5 = 19,4

Thời gian hoàn thành:

P1 = 10, P2 = 61 -2 = 59, P3 = 23 - 4 = 19, P4 = 30 – 5 = 25 , P5 = 52 - 7 = 45

* Thời gian chờ trung bình: (10 + 59 + 19 + 25 + 45)/5 = 31,6

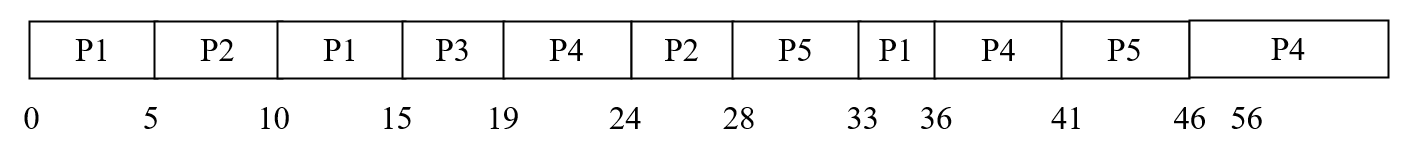
A screen shot of a computer

Description automatically generated

RR (quantum time = 5)

|  |  |  |
| --- | --- | --- |
| **Process** | **Arrival Time** | **Burst Time** |
| P1 | 0 | 13 |
| P2 | 4 | 9 |
| P3 | 6 | 4 |
| P4 | 7 | 20 |
| P5 | 12 | 10 |

Kết quả giản đồ gantt:



Thời gian đợi trung bình: ((5 + 18) + (1 + 14) + 9 + (12 + 12 + 5) + (16 + 8))/5 = 20

Thời gian hoàn thành trung bình: (36 + 24 + 13 + 49 + 34)/5 = 31.2

A screen shot of a computer

Description automatically generated

Code:

#include<stdio.h>

int pn[10], arr[10], arr1[10], bur[10], bur1[10], start[10], finish[10], tat[10], wt[10];

void hv(int a, int b)

{

    int tmp = pn[a];

    pn[a] = pn[b];

    pn[b] = tmp;

    tmp = arr[a];

    arr[a] = arr[b];

    arr[b] = tmp;

    tmp = arr1[a];

    arr1[a] = arr1[b];

    arr1[b] = tmp;

    tmp = bur[a];

    bur[a] = bur[b];

    bur[b] = tmp;

    tmp = bur1[a];

    bur1[a] = bur1[b];

    bur1[b] = tmp;

    tmp = start[a];

    start[a] = start[b];

    start[b] = tmp;

    tmp = finish[a];

    finish[a] = finish[b];

    finish[b] = tmp;

}

void Sort(int b)

{

    int i, j;

    for(i = 0; i < b; i++)

        for(j = i + 1; j <= b; j++)

            if(arr1[i] > arr1[j])

                hv(i, j);

            else if((arr1[i] == arr1[j]) && (bur1[i] > bur1[j]))

                hv(i, j);

    for(i = b; i >= 1; i--)

        if((arr1[i] == arr1[i-1]) && (arr[i] == arr1[i]) && (arr[i-1] == arr1[i - 1]))

            hv(i, i - 1);

}

int main()

{

    int totwt = 0, tottat = 0, i, n, j, QTime;

    printf("Enter the number of processes: ");

    scanf("%d", &n);

    printf("Enter QTime: ");

    scanf("%d", &QTime);

    for (i = 0; i < n; i++)

    {

        printf("Enter the Process Name, Arrival Time, Burst Time: ");

        scanf("%d%d%d", &pn[i], &arr[i], &bur[i]);

        arr1[i] = arr[i];

        bur1[i] = bur[i];

    }

    for(i = 0; i < n -1; i++)

        for(j = i + 1; j < n; j++)

        {

            if(arr[i] > arr[j])

                hv(i, j);

            else if((arr[i] == arr[j]) && (bur[i] < bur[j]))

                    hv(i, j);

        }

    int PFinish = 0, time = 0;

    while(PFinish < n)

    {

        while(arr1[0] > time)

            time++;

        if(arr[0] == arr1[0])

            start[0] = time;

        if(bur1[0] <= QTime)

        {

            time = time + bur1[0];

            finish[0] = time;

            hv(0, n-1-PFinish);

            PFinish++;

            Sort(n-1-PFinish);

        }

        else

        {

            time = time + QTime;

            bur1[0] = bur1[0] - QTime;

            arr1[0] = time;

            Sort(n-1-PFinish);

        }

    }

    printf("\nPName Arrtime Burtime Start Tat Finish Wt");

    for(i = 0; i < n; i++)

    {

        tat[i] = finish[i] - arr[i];

        wt[i] = tat[i] - bur[i];

        printf("\nP%d\t%6d\t%6d\t%6d\t%6d\t%6d\t%6d", pn[i], arr[i], bur[i], start[i], tat[i], finish[i], wt[i]);

        totwt += wt[i];

        tottat += tat[i];

    }

    printf("\nAvgwt= %f\nAvgtat= %f\n", (float)totwt / n, (float)tottat / n);

return 0;

}