Code :

#include<stdio.h>

int main()

{

    int ready\_queue[100]={1,2};

    int arrival\_t[100]={0,0};

    int burst\_t[100]={10,7};

    int ready\_priority[100]={2,3};

    int time\_quantum=4;

    int gantt\_proccesses[100]={};

    int gantt\_time[100]={};

    int high\_priority;

    int current\_proccess=0;

    int new\_process=0;

    int run\_for=0;

    int burst\_replica[100]={10,7};

    int tot\_run\_completed[100]={};

    int no\_of\_sig[100]={};

    int shift\_to\_pos=0;

    int proc\_counter=2;

    while(1)

    {

        //check for smallest number i.e highest priority

        AGAIN:high\_priority=ready\_priority[0];

        current\_proccess=0;

        for(int i=1;i<100;i++)

        {

            if ((ready\_priority[i] < high\_priority) && (ready\_priority[i] != 0))

            {

                high\_priority = ready\_priority[i];

                current\_proccess = i;

            }

        }

        //for exit

        if (ready\_priority[current\_proccess] == 10)

            break;

        //for new process when burst is done

        if (burst\_t[current\_proccess] == 0)

        {

            ready\_priority[current\_proccess] = 10;

            goto AGAIN;

        }

        //display

        //printf("Priority is %d \n",high\_priority);

        //printf("Process is %d \n",ready\_queue[current\_proccess]);

        //add in gantt for 1st process

        if (gantt\_proccesses[0] == 0)

        {

            //gantt\_proccesses[0] = ready\_queue[current\_proccess];

            gantt\_time[0]       = arrival\_t[current\_proccess];

        }

        //adding process in gantt

        for(int i=0;i<100;i++)

        {

            if (gantt\_proccesses[i] == 0)

            {

                gantt\_proccesses[i] = ready\_queue[current\_proccess];

                break;

            }

        }

        //check burst with time quant

        if (burst\_t[current\_proccess] >= time\_quantum)

        {

            burst\_t[current\_proccess] -= time\_quantum;

            run\_for = time\_quantum;

        }

        else

        {

            run\_for = burst\_t[current\_proccess];

            burst\_t[current\_proccess] = 0;

        }

        tot\_run\_completed[current\_proccess] += run\_for;

        //adding time in gantt

        for(int i=1;i<100;i++)

        {

            if (gantt\_time[i] == 0)

            {

                gantt\_time[i] = gantt\_time[i-1] + run\_for;

                break;

            }

        }

        //adding new proc after every 3 ticks

        //if((run\_for >= 3) && (ready\_priority[current\_proccess] != 1))

        if((tot\_run\_completed[current\_proccess] >= 3) && (ready\_priority[current\_proccess] != 1))

        {

            tot\_run\_completed[current\_proccess] -= 3;

            //add new proc in ready queue

            for(int i=0;i<50;i++)

            {

                if(ready\_queue[i] == 0)

                {

                    ready\_queue[i] = proc\_counter + 1;

                    proc\_counter += 1;

                    new\_process = i;

                    if(ready\_priority[current\_proccess] == 2)

                    {

                        ready\_priority[new\_process] = 3;

                        burst\_t[new\_process] = 7;

                        burst\_replica[new\_process] = 7;

                    }

                    if(ready\_priority[current\_proccess] == 3)

                    {

                        ready\_priority[new\_process] = 1;

                        burst\_t[new\_process] = 5;

                        burst\_replica[new\_process] = 5;

                    }

                    //printf("new process %d ",ready\_queue[i]);

                    //printf("new process priority %d ",ready\_priority[new\_process]);

                    //printf("new process burst %d \n",burst\_t[new\_process]);

                    break;

                }

            }

        }

        //to delete current process and put ahead if burst remains

        if(burst\_t[current\_proccess] != 0)

        {

            for(int i=0;i<50;i++)

            {

                if(ready\_priority[i] == 0)

                {

                    shift\_to\_pos = i;

                    break;

                }

            }

            //printf("\n %d shifted to %d\n",ready\_queue[current\_proccess],shift\_to\_pos);

            burst\_t[shift\_to\_pos]=burst\_t[current\_proccess];

            ready\_queue[shift\_to\_pos]=ready\_queue[current\_proccess];

            ready\_priority[shift\_to\_pos]=ready\_priority[current\_proccess];

            tot\_run\_completed[shift\_to\_pos]=tot\_run\_completed[current\_proccess];

        }

        //printf("\ndeleted from %d\n",current\_proccess);

        for(int i=current\_proccess;i<50;i++)

        {

            burst\_t[i]=burst\_t[i+1];

            ready\_queue[i]=ready\_queue[i+1];

            ready\_priority[i]=ready\_priority[i+1];

            tot\_run\_completed[i]=tot\_run\_completed[i+1];

        }

    }

    //print ready queue

    //for(int g=0;g<20;g++)

    //{

    //    printf("%d ",ready\_queue[g]);

    //    if (ready\_queue[g+1] == 0)

    //        break;

    //}

    //print ready priority

    //printf("\n");

    //for(int g=0;g<20;g++)

    //{

    //    printf("%d ",ready\_priority[g]);

    //    if (ready\_priority[g+1] == 0)

    //        break;

    //}

    //print burst

    //printf("\n");

    //for(int g=0;g<13;g++)

    //{

    //    printf("%d ",burst\_replica[g]);

    //}

    //for number of signals

    for(int i=3;i<100;i=i+3)

    {

        for(int j=0;j<100;j++)

        {

            if((gantt\_time[j]<=i) && (i<gantt\_time[j+1]))

            {

                no\_of\_sig[gantt\_proccesses[j]] += 1;

            }

        }

    }

    printf("\nThe number of Signals received by each process is as follows : \n");

    for(int i=1;i<14;i++)

    {

        if(i>9)

            printf("P%d : %d \n",i,no\_of\_sig[i]);

        else

            printf("P%d  : %d \n",i,no\_of\_sig[i]);

    }

    printf("\nThe Gantt Chart is as follows : \n");

    //print gantt chart

    printf("-------------------------------------------------------------------------------------------------------------");

    printf("\n");

    printf("|");

    for(int g=0;g<50;g++)

    {

        if(gantt\_proccesses[g] > 9)

            printf("P%d|",gantt\_proccesses[g]);

        else

            printf(" P%d|",gantt\_proccesses[g]);

        if (gantt\_proccesses[g+1] == 0)

            break;

    }

    printf("\n-------------------------------------------------------------------------------------------------------------");

    printf("\n");

    for(int g=0;g<50;g++)

    {

        if(gantt\_time[g] > 9)

            printf("%d  ",gantt\_time[g]);

        else

            printf(" %d  ",gantt\_time[g]);

        if (gantt\_time[g+1] == 0)

            break;

    }

    return 0;

}