```c
// SJF (Preemptive)

#include <stdio.h>

struct pack
{
    int id;
    int AT, BT, WT, TA, BTT;
    int r;
};

void main()
{

    struct pack arr[100];
    int totalbt = 0;
    int n;
    printf("Enter the count of Processes: ");
    scanf("%d", &n);
    for (int i = 0; i < n; i++)
    {
        printf("Enter for %d:\n", i);
        arr[i].id = i;
        arr[i].r = 0;
        arr[i].TA = 0;
        arr[i].WT = 0;

        printf("AT: ");
        scanf("%d", &arr[i].AT);
        printf("BT: ");
        scanf("%d", &arr[i].BT);
        arr[i].BTT = arr[i].BT;
        totalbt += arr[i].BT;
    }

    for (int i = 0; i < n - 1; i++)
    {
        for (int j = 0; j < n - i - 1; j++)
        {
            if (arr[j].AT > arr[j + 1].AT)
            {
                struct pack temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
            else if (arr[j].AT == arr[j + 1].AT)
            {
                if (arr[j].BT > arr[j + 1].BT)
```

```
            {
                struct pack temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
}
printf("\n Gantt Chart:\n\n");

int CT = 0, i = 0;
int onetime = 0;
int sin = 0;
float avgTA = 0, pre = -1;
if (arr[0].AT != 0)
{
    CT = arr[0].AT;
    printf("0--///--");
}
while (totalbt > 0)
{
    int flag = 0;
    if (i < n && CT == arr[i].AT)
    {
        arr[i].r = 1;
        if (arr[sin].BT < 1 || arr[sin].BT > arr[i].BT)
        {
            sin = i;
        }
        i++;
    }
    if (arr[sin].BT > 0)
    {
        arr[sin].BT--;
        totalbt--;
        flag = 1;
        onetime = 0;

        if (pre != sin)
        {
            pre = sin;
            printf("%d--P%d--", CT, arr[sin].id);
        }
        for (int p = 0; p < n; p++)
        {
            if (arr[p].r == 1)
            {
                arr[p].TA++;
```

```
                avgTA++;
            }
        }
    }

    if (arr[sin].BT < 1)
    {
        arr[sin].r = 0;
        int k;
        for (k = 0; k < n; k++)
        {
            if (arr[k].r == 1)
            {
                sin = k;
                break;
            }
        }
        if (k == n && flag == 0 && onetime == 0)
        {
            printf("%d--///--", CT);
            flag = 1;
            onetime = 1;
        }
        for (int k = 0; k < n; k++)
        {
            if (arr[k].r == 1)
            {
                if (arr[sin].BT > arr[k].BT)
                {
                    sin = k;
                }
            }
        }
    }
    CT++;
}
printf("%d", CT);
float avgWt = 0;
for (int p = 0; p < n; p++)
{
    arr[p].WT = arr[p].TA - arr[p].BTT;
    avgWt += arr[p].WT;
}

printf("\n\nPid     AT     BT     TA     WT\n");
for (int i = 0; i < n; i++)
{
```

```
        printf("P%d      %d      %d      %d      %d\n", arr[i].id, arr[i].AT,
arr[i].BT, arr[i].TA, arr[i].WT);
    }

    printf("\nAverage Turn Around Time: %.2f\n", avgTA / (float)(n));
    printf("Average Waiting Time: %.2f\n ", avgWt / (float)(n));
}
```

// RR

```c
#include<stdio.h>
 #include<conio.h>

 void main()
 {

 // initlialize the variable name
 int i, NOP, sum=0,count=0, y, quant, wt=0, tat=0, at[10], bt[10], temp[10];
 float avg_wt, avg_tat;
 printf(" Total number of process in the system: ");
 scanf("%d", &NOP);
 y = NOP; // Assign the number of process to variable y

 // Use for loop to enter the details of the process like Arrival time and the
Burst Time
 for(i=0; i<NOP; i++)
 {
 printf("\n Enter the Arrival and Burst time of the Process[%d]\n", i+1);
 printf(" Arrival time is: \t"); // Accept arrival time
 scanf("%d", &at[i]);
 printf(" \nBurst time is: \t"); // Accept the Burst time
 scanf("%d", &bt[i]);
 temp[i] = bt[i]; // store the burst time in temp array
 }
 // Accept the Time qunat
 printf("Enter the Time Quantum for the process: \t");
 scanf("%d", &quant);
 // Display the process No, burst time, Turn Around Time and the waiting time
 printf("\n Process No \t\t Burst Time \t\t TAT \t\t Waiting Time ");
 for(sum=0, i = 0; y!=0; )
 {
 if(temp[i] <= quant && temp[i] > 0) // define the conditions
 {
 sum = sum + temp[i];
 temp[i] = 0;
 count=1;
 }
 else if(temp[i] > 0)
 {
 temp[i] = temp[i] - quant;
 sum = sum + quant;
 }
 if(temp[i]==0 && count==1)
 {
 y--; //decrement the process no.
```

```
 printf("\nProcess No[%d] \t\t %d\t\t\t\t %d\t\t\t %d", i+1, bt[i], sum-at[i],
sum-at[i]-bt[i]);
 wt = wt+sum-at[i]-bt[i];
 tat = tat+sum-at[i];
 count =0;
 }
 if(i==NOP-1)
 {
 i=0;
 }
 else if(at[i+1]<=sum)
 {
 i++;
 }
 else
 {
 i=0;
 }
 }
 // represents the average waiting time and Turn Around time
 avg_wt = wt * 1.0/NOP;
 avg_tat = tat * 1.0/NOP;
 printf("\n Average Turn Around Time: \t%f", avg_wt);
 printf("\n Average Waiting Time: \t%f", avg_tat);
 getch();
 }
```