Implementation of RR (Round Robin) CPU Scheduling Algorithm

Aim:

To schedule snapshot of processes queued according to round robin scheduling.

Round Robin

- All processes are processed one by one as they have arrived but in rounds.
- Each process cannot take more than the time slice per round.
- Round Robin is a fair preemptive scheduling algorithm.
- A process that is yet to complete in a round is preempted after the time slice and put at the end of the queue.
- When a process is completely processed, it is removed from the queue.

Algorithm:

- Get length of the ready queue.
- Obtain Burst time Bi for each processes Pi
- Get the time slice per round, say TS
- Determine the number of rounds for each process.
- The wait time for first process is 0.
- If Bi > TS then process takes more than one round. Therefore turnaround and waiting time should include the time spent for other

Remaining processes in the same round.

- Calculate average waiting time and turn around time.
- Display the GANTT chart that includes.
  a. order in which the processes were processed in progression of rounds.
  b. Turnaround time Ti for each process in progression of rounds.
- Display the burst time, turnaround time and wait time for each process.
- Display average wait time and turnaround time
- Stop.

IMPLEMENTATION OF ROUND ROBIN CPU SCHEDULING ALGORITHM

PROGRAM:

```
#include <stdio.h>
int main()
{
    int i,x=-1,k[10],m=0,n,t,s=0;
    int a[50],temp,b[50],p[10],bur[10],bur1[10];
    int wat[10],tur[10],ttur=0,twat=0 j=0;
    float awat,atur;
    printf("Enter no. of process : ");
    scanf("%d", &n);
    for(i=0; i<n; i++)
    {
        printf("Burst time for process P%d : ", (i+1));
        scanf("%d", &bur[i]);
        bur1[i] = bur[i];
    }
    printf("Enter the time slice (in ms) : ");
    scanf("%d", &t);
    for(i=0; i<n; i++)
    {
        b[i] = bur[i] / t;
        if((bur[i]%t) != 0)
        b[i] += 1;
        m += b[i];
```

```
}
printf("\n\t\tRound Robin Scheduling\n");
printf("\nGANTT Chart\n");
for(i=0; i<m; i++)
        printf("--------");
printf("\n");
a[0] = 0;
while(j < m)
{
        if(x == n-1)
        x = 0;
        else
        x++;
        if(bur[x] >= t)
        {
                bur[x] -= t;
                a[j+1] = a[j] + t;
                if(b[x] == 1)
                {
                        p[s] = x;
                        k[s] = a[j+1];
                        s++;
                }
                j++;
                b[x] -= 1;
                printf(" P%d|", x+1);
}
```

```
            else if(bur[x] != 0)
            {
                    a[j+1] = a[j] + bur[x];
                    bur[x] = 0;
                    if(b[x] == 1)
                    {
                            p[s] = x;
                            k[s] = a[j+1];
                            s++;
                    }
                    j++;
                    b[x] -= 1;
                    printf(" P%d |",x+1);
            }
    }
    printf("\n");
    for(i=0;i<m;i++)
            printf("--------");
    printf("\n");
    for(j=0; j<=m; j++)
            printf("%d\t", a[j]);
    for(i=0; i<n;i++)
    {
            for(j=i+1;j<n: i++)
            {
                            p[j])
                    {
```

```
                    temp= p[i];
                    p[i]= p[j];
                    p[j]= temp;
                    temp = k[i];
                    k[i] = k[j];
                    k[j] = temp;
                }
            }
        }
        for(i=0; i<n; i++)
        {
            wat[i] = k[i] - bur1[i];
            tur[i] = k[i];
        }
        for(i=0; i<n; i++)
        {
            ttur += tur[i];
            twat += wat[i];
        }
        printf("\n\n");
        for(i=0; i<30; i++)
            printf("-");
        printf("\nProcess\tBurst\tTrnd\tWait\n");
        for(i=0; i<30; i++)
            printf("-");
        for (i=0; i<n; i++)
            printf("\nP%-4d\t%4d\t%4d\t%4d", p[i]+1, bur1[i],tur[i],wat[i]);
```

```
printf("\n");
for(i=0; i<30; i++)
        printf("-");
awat = (float)twat / n;
atur = (float)ttur / n;
printf("\n\nAverage waiting time    :%.2f ms", awat);
printf("\nAverage turn around time : %.2f ms\n", atur);
}
```

Result

Thus waiting time and turn around time for processes based on round robin scheduling was computed and the average waiting time was determined.