Operating System – CS23431

Ex 6d)	**ROUND ROBIN SCHEDULING**
Name: B M Madhumitha	
Reg No: 230701168	

Aim:

To implement the Round Robin (RR) scheduling technique

Algorithm:

1. Declare the structure and its elements.
2. Get number of processes and Time quantum as input from the user.
3. Read the process name, arrival time and burst time
4. Create an array **rem_bt[]** to keep track of remaining burst time of processes which is initially copy of bt[] (burst times array)
5. Create another array **wt[]** to store waiting times of processes. Initialize this array as 0. 6. Initialize time : t = 0
7. Keep traversing the all processes while all processes are not done. Do following for i'th process if it is not done yet.

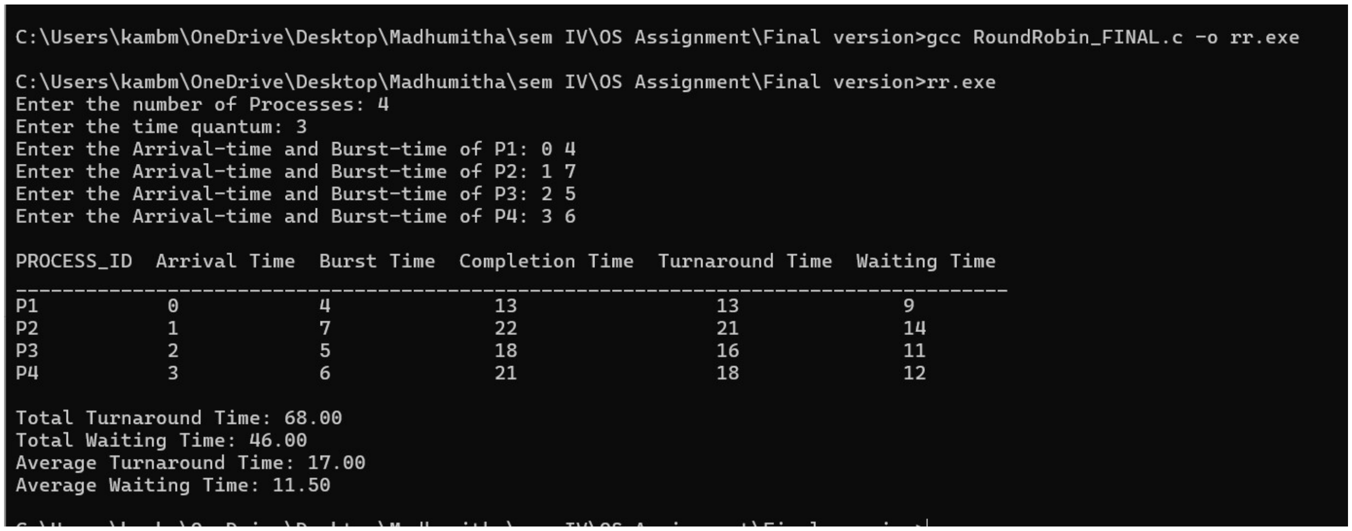
```
a- If rem_bt[i] > quantum
(i) t = t + quantum
(ii) bt_rem[i] -= quantum;
b- Else // Last cycle for this process
(i) t = t + bt_rem[i];
(ii) wt[i] = t - bt[i]
(iii) bt_rem[i] = 0; // This process is over
```

8. Calculate the waiting time and turnaround time for each process.
9. Calculate the average waiting time and average turnaround time.
10. Display the results.

Program Code:

```
#include <stdio.h>

void sort(int bt[], int at[], int n, int p[]) {
    for (int i = 0; i < n - 1; i++) {
        for (int j = i + 1; j < n; j++) {
            if (at[j] < at[i]) {
                // Swap Arrival Time
                int temp = at[j];
                at[j] = at[i];
                at[i] = temp;

                // Swap Burst Time
                temp = bt[i];
                bt[i] = bt[j];
                bt[j] = temp;

                // Swap Process ID
                temp = p[i];
                p[i] = p[j];
                p[j] = temp;
            }
        }
    }
}

void display(int n, int at[], int bt[], int ct[], int tat[], int wt[], int p[]) {
    printf("\nPROCESS_ID  Arrival Time  Burst Time  Completion Time  Turnaround Time  Waiting Time\n");
    printf("__________________________________________________________________________________________
__\n");
    for (int i = 0; i < n; i++) {
        printf("P%d          %d           %d             %d                %d             %d\n",
               p[i], at[i], bt[i], ct[i], tat[i], wt[i]);
    }
}

void average(int n, int tat[], int wt[]) {
    float total_tat = 0, total_wt = 0;
    for (int i = 0; i < n; i++) {
        total_tat += tat[i];
        total_wt += wt[i];
    }
    printf("\nTotal Turnaround Time: %.2f\nTotal Waiting Time: %.2f\n", total_tat, total_wt);
    printf("Average Turnaround Time: %.2f\nAverage Waiting Time: %.2f\n", total_tat / n, total_wt / n);
}

int main() {
    int quant, n;
    printf("Enter the number of Processes: ");
    scanf("%d", &n);
    printf("Enter the time quantum: ");
    scanf("%d", &quant);

    int bt[n], at[n], rem_bt[n], wt[n], p[n];
    int ct[n], tat[n];

    for (int i = 0; i < n; i++) {
        printf("Enter the Arrival-time and Burst-time of P%d: ", i + 1);
        scanf("%d %d", &at[i], &bt[i]);
        rem_bt[i] = bt[i];

        wt[i] = 0;
        p[i] = i + 1;
    }
    sort(bt, at, n, p);

    int t = at[0]; // Start time is the first process's arrival
    int count = 0;

    while (count != n) {
        int executed = 0;

        // Process only those which have arrived up to time 't'
        for (int i = 0; i < n; i++) {
            if (rem_bt[i] > 0 && at[i] <= t) {
                executed = 1;

                if (rem_bt[i] > quant) {
                    t += quant;
                    rem_bt[i] -= quant;
                } else {
                    t += rem_bt[i];
                    ct[i] = t;
                    tat[i] = ct[i] - at[i];  // Turnaround Time
                    wt[i] = tat[i] - bt[i];  // Waiting Time
                    rem_bt[i] = 0;
                    count++;
                }
            }
            else if(at[i]>t){
                break;
            }
        }

        // If no process was executed, move `t` to the next available process
        if (executed == 0) {
            for (int i = 0; i < n; i++) {
                if (rem_bt[i] > 0) {
                    t = at[i];
                    break;
                }
            }
        }
    }

    display(n, at, bt, ct, tat, wt, p);
    average(n, tat, wt);

    return 0;
}
```

Output:



```
C:\Users\kambm\OneDrive\Desktop\Madhumitha\sem IV\OS Assignment\Final version>gcc RoundRobin_FINAL.c -o rr.exe

C:\Users\kambm\OneDrive\Desktop\Madhumitha\sem IV\OS Assignment\Final version>rr.exe
Enter the number of Processes: 4
Enter the time quantum: 3
Enter the Arrival-time and Burst-time of P1: 0 4
Enter the Arrival-time and Burst-time of P2: 1 7
Enter the Arrival-time and Burst-time of P3: 2 5
Enter the Arrival-time and Burst-time of P4: 3 6

PROCESS_ID  Arrival Time  Burst Time  Completion Time  Turnaround Time  Waiting Time
---------------------------------------------------------------------------------------
P1          0             4             13                 13               9
P2          1             7             22                 21               14
P3          2             5             18                 16               11
P4          3             6             21                 18               12

Total Turnaround Time: 68.00
Total Waiting Time: 46.00
Average Turnaround Time: 17.00
Average Waiting Time: 11.50
```

Result: Thus, the program was successfully executed.