Name: Nikhil Gupta
Roll No.: 20051523

LAB ASSIGNMENT 11

Round Robin Scheduling

```c
#include <stdio.h>
#include <limits.h>
#include <stdbool.h>
#include <stdlib.h>

struct process_struct
{
    int pid;
    int at;
    int bt;
    int ct, wt, tat, rt, start_time;
    int bt_remaining;
} ps[100];

int pfline(int num)
{
    for (int i = 0; i < num; i++)
    {
        printf("-");
    }
    printf("\n");
}
int findmax(int a, int b)
{
    return a > b ? a : b;
}

int comparatorAT(const void *a, const void *b)
{
    int x = ((struct process_struct *)a)->at;
    int y = ((struct process_struct *)b)->at;
    if (x < y)
```

Name: Nikhil Gupta
Roll No.: 20051523

```
        return -1;
    else if (x >= y)
        return 1;
}

int comparatorPID(const void *a, const void *b)
{
    int x = ((struct process_struct *)a)->pid;
    int y = ((struct process_struct *)b)->pid;
    if (x < y)
        return -1;
    else if (x >= y)
        return 1;
}

int main()
{

    int n, index;
    int cpu_utilization;

    bool visited[100] = {false}, is_first_process = true;
    int current_time = 0, max_completion_time;
    int completed = 0, tq, total_idle_time = 0,
length_cycle;
    printf("Enter total number of processes: ");
    scanf("%d", &n);
    int queue[100], front = -1, rear = -1;
    float sum_tat = 0, sum_wt = 0, sum_rt = 0;

    for (int i = 0; i < n; i++)
    {
        printf("\nEnter Process %d Arrival Time: ", i);
        scanf("%d", &ps[i].at);
        ps[i].pid = i;
    }

    for (int i = 0; i < n; i++)
    {
        printf("\nEnter Process %d Burst Time: ", i);
```

```
        scanf("%d", &ps[i].bt);
        ps[i].bt_remaining = ps[i].bt;
    }

    printf("\nEnter time quantum: ");
    scanf("%d", &tq);

    qsort((void *)ps, n, sizeof(struct process_struct),
comparatorAT);

    front = rear = 0;
    queue[rear] = 0;
    visited[0] = true;

    while (completed != n)
    {
        index = queue[front];

        front++;

        if (ps[index].bt_remaining == ps[index].bt)
        {
            ps[index].start_time = findmax(current_time,
ps[index].at);
            total_idle_time += (is_first_process == true)
? 0 : ps[index].start_time - current_time;
            current_time = ps[index].start_time;
            is_first_process = false;
        }

        if (ps[index].bt_remaining - tq > 0)
        {
            ps[index].bt_remaining -= tq;
            current_time += tq;
        }
        else
        {
            current_time += ps[index].bt_remaining;
            ps[index].bt_remaining = 0;
            completed++;
```

Name: Nikhil Gupta
Roll No.: 20051523

```
            ps[index].ct = current_time;
            ps[index].tat = ps[index].ct - ps[index].at;
            ps[index].wt = ps[index].tat - ps[index].bt;
            ps[index].rt = ps[index].start_time -
ps[index].at;

            sum_tat += ps[index].tat;
            sum_wt += ps[index].wt;
            sum_rt += ps[index].rt;
        }

        for (int i = 1; i < n; i++)
        {
            if (ps[i].bt_remaining > 0 && ps[i].at <=
current_time && visited[i] == false)
            {

                queue[++rear] = i;
                visited[i] = true;
            }
        }

        if (ps[index].bt_remaining > 0)

            queue[++rear] = index;

        if (front > rear)
        {
            for (int i = 1; i < n; i++)
            {
                if (ps[i].bt_remaining > 0)
                {
                    queue[rear++] = i;
                    visited[i] = true;
                    break;
                }
            }
        }
    }
```

Name: Nikhil Gupta
Roll No.: 20051523

```
    max_completion_time = INT_MIN;

    for (int i = 0; i < n; i++)
        max_completion_time =
findmax(max_completion_time, ps[i].ct);

    length_cycle = max_completion_time - ps[0].at;

    cpu_utilization = (float)(length_cycle -
total_idle_time) / length_cycle;

    qsort((void *)ps, n, sizeof(struct process_struct),
comparatorPID);

    printf("\nProcess No.\tAT\tCPU Burst Time\tStart
Time\tCT\tTAT\tWT\tRT\n");
    pfline(90);
    for (int i = 0; i < n; i++)
        printf("%d\t\t%d\t%d\t\t%d\t\t%d\t%d\t%d\t%d\n",
i, ps[i].at, ps[i].bt, ps[i].start_time, ps[i].ct,
ps[i].tat, ps[i].wt, ps[i].rt);
    printf("\n");

    printf("\nAverage Turn Around time= %.2f",
(float)sum_tat / n);
    printf("\nAverage Waiting Time= %.2f", (float)sum_wt
/ n);
    printf("\nAverage Response Time= %.2f", (float)sum_rt
/ n);
    return 0;
}
```

Name: Nikhil Gupta
Roll No.: 20051523

OUTPUT

```
Enter total number of processes: 5

Enter Process 0 Arrival Time: 0

Enter Process 1 Arrival Time: 1

Enter Process 2 Arrival Time: 3

Enter Process 3 Arrival Time: 5

Enter Process 4 Arrival Time: 6

Enter Process 0 Burst Time: 8

Enter Process 1 Burst Time: 6

Enter Process 2 Burst Time: 3

Enter Process 3 Burst Time: 2

Enter Process 4 Burst Time: 4

Enter time quantum: 4

Process No.     AT      CPU Burst Time  Start Time      CT      TAT     WT      RT
-----------------------------------------------------------------------------------
0               0       8               0               15      15      7       0
1               1       6               4               23      22      16      3
2               3       3               8               11      8       5       5
3               5       2               15              17      12      10      10
4               6       4               17              21      15      11      11


Average Turn Around time= 14.40
Average Waiting Time= 9.80
Average Response Time= 5.80
PS E:\Mega Sync\Programming\C\Scheduling Algorithms>
```