Name – Harshit

MIS – 112316018

OS Lab

Assignment 5B

To write a C program for implementation of Round Robin scheduling algorithms

Code:

#include <stdio.h>

#include <stdlib.h>

struct Process {

    int pid;

    int burst\_time;

    int arrival\_time;

    int remaining\_time;

    int completion\_time;

    int turnaround\_time;

    int waiting\_time;

};

struct GanttEntry {

    int pid;

    int start\_time;

    int end\_time;

};

void calculateTimesWithGantt(struct Process processes[], int n, int quantum,

                           struct GanttEntry gantt[], int \*gantt\_size) {

    int time = 0;

    int done = 0;

    \*gantt\_size = 0;

    for(int i = 0; i < n; i++) {

        processes[i].remaining\_time = processes[i].burst\_time;

    }

    while(done < n) {

        for(int i = 0; i < n; i++) {

            if(processes[i].remaining\_time > 0) {

                gantt[\*gantt\_size].pid = processes[i].pid;

                gantt[\*gantt\_size].start\_time = time;

                if(processes[i].remaining\_time > quantum) {

                    time += quantum;

                    processes[i].remaining\_time -= quantum;

                }

                else {

                    time += processes[i].remaining\_time;

                    processes[i].completion\_time = time;

                    processes[i].remaining\_time = 0;

                    done++;

                    processes[i].turnaround\_time = processes[i].completion\_time - processes[i].arrival\_time;

                    processes[i].waiting\_time = processes[i].turnaround\_time - processes[i].burst\_time;

                }

                gantt[\*gantt\_size].end\_time = time;

                (\*gantt\_size)++;

            }

        }

    }

}

void displayGanttChart(struct GanttEntry gantt[], int size) {

    printf("\nGantt Chart:\n");

    printf(" ");

    for(int i = 0; i < size; i++) {

        for(int j = 0; j < 6; j++) printf("-");

        printf(" ");

    }

    printf("\n");

    printf("|");

    for(int i = 0; i < size; i++) {

        printf(" P%d  |", gantt[i].pid);

    }

    printf("\n");

    printf(" ");

    for(int i = 0; i < size; i++) {

        for(int j = 0; j < 6; j++) printf("-");

        printf(" ");

    }

    printf("\n");

    printf("0");

    for(int i = 0; i < size; i++) {

        if(gantt[i].end\_time < 10)

            printf("      %d", gantt[i].end\_time);

        else

            printf("     %d", gantt[i].end\_time);

    }

    printf("\n");

}

void displayResults(struct Process processes[], int n) {

    float avg\_waiting\_time = 0, avg\_turnaround\_time = 0;

    printf("\nProcess\tBurst Time\tArrival Time\tCompletion Time\tTurnaround Time\tWaiting Time\n");

    for(int i = 0; i < n; i++) {

        printf("%d\t%d\t\t%d\t\t%d\t\t%d\t\t%d\n",

               processes[i].pid,

               processes[i].burst\_time,

               processes[i].arrival\_time,

               processes[i].completion\_time,

               processes[i].turnaround\_time,

               processes[i].waiting\_time);

        avg\_waiting\_time += processes[i].waiting\_time;

        avg\_turnaround\_time += processes[i].turnaround\_time;

    }

    avg\_waiting\_time /= n;

    avg\_turnaround\_time /= n;

    printf("\nAverage Waiting Time: %.2f", avg\_waiting\_time);

    printf("\nAverage Turnaround Time: %.2f\n", avg\_turnaround\_time);

}

int main() {

    int n, quantum;

    printf("Enter the number of processes: ");

    scanf("%d", &n);

    struct Process processes[n];

    struct GanttEntry \*gantt = malloc(sizeof(struct GanttEntry) \* 100);

    int gantt\_size;

    printf("Enter time quantum: ");

    scanf("%d", &quantum);

    for(int i = 0; i < n; i++) {

        processes[i].pid = i + 1;

        printf("\nFor Process %d:\n", i + 1);

        printf("Enter Burst Time: ");

        scanf("%d", &processes[i].burst\_time);

        printf("Enter Arrival Time: ");

        scanf("%d", &processes[i].arrival\_time);

    }

    calculateTimesWithGantt(processes, n, quantum, gantt, &gantt\_size);

    displayResults(processes, n);

    displayGanttChart(gantt, gantt\_size);

    free(gantt);

    return 0;

}

Output:

