Name: Vanshikha Vijaysingh Panwar

PRN: 2301106068

1)Shortest Job First (SJF):

* Input:

#include <stdio.h>

#include <stdlib.h>

// Structure to represent a process

struct Process {

int arrival\_time;

int burst\_time;

int completion\_time;

int waiting\_time;

};

// Function to perform SJF scheduling

void SJF(struct Process\* processes, int n) {

// Sort processes based on burst time

for (int i = 0; i < n - 1; ++i) {

for (int j = 0; j < n - i - 1; ++j) {

if (processes[j].burst\_time > processes[j + 1].burst\_time) {

// Swap processes

struct Process temp = processes[j];

processes[j] = processes[j + 1];

processes[j + 1] = temp;

}

}

}

// Calculate completion time and waiting time for each process

int current\_time = 0;

for (int i = 0; i < n; ++i) {

processes[i].completion\_time = current\_time + processes[i].burst\_time;

processes[i].waiting\_time = current\_time - processes[i].arrival\_time;

current\_time += processes[i].burst\_time;

}

// Calculate total waiting time and turnaround time

double total\_waiting\_time = 0, total\_turnaround\_time = 0;

for (int i = 0; i < n; ++i) {

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

total\_turnaround\_time += processes[i].completion\_time - processes[i].arrival\_time;

}

double average\_waiting\_time = total\_waiting\_time / n;

double average\_turnaround\_time = total\_turnaround\_time / n;

// Display results

printf("Process\tBurst Time\tCompletion Time\tWaiting Time\n");

for (int i = 0; i < n; ++i) {

printf("P%d\t\t%d\t\t%d\t\t%d\n", i+1, processes[i].burst\_time,

processes[i].completion\_time, processes[i].waiting\_time);

}

printf("\nAverage Waiting Time: %.2lf\n", average\_waiting\_time);

printf("Average Turnaround Time: %.2lf\n", average\_turnaround\_time);

}

int main() {

// Example usage

struct Process processes[] = { {0, 6}, {1, 8}, {2, 7}, {3, 3} };

int n = sizeof(processes) / sizeof(processes[0]);

SJF(processes, n);

return 0;

}

* Output:

Process Burst Time Completion Time Waiting Time

P1 3 3 -3

P2 6 9 3

P3 7 16 7

P4 8 24 15

Average Waiting Time: 5.50

Average Turnaround Time: 11.50

2)Priority Scheduling:

* Input:

#include <stdio.h>

#include <stdlib.h>

// Structure to represent a process

struct Process {

int process\_id;

int arrival\_time;

int burst\_time;

int priority;

int completion\_time;

};

// Function to perform Priority Scheduling and display Gantt chart

void priorityScheduling(struct Process\* processes, int n) {

// Sort processes based on priority (higher priority first)

for (int i = 0; i < n - 1; ++i) {

for (int j = 0; j < n - i - 1; ++j) {

if (processes[j].priority < processes[j + 1].priority) {

// Swap processes

struct Process temp = processes[j];

processes[j] = processes[j + 1];

processes[j + 1] = temp;

}

}

}

int current\_time = 0;

printf("Gantt Chart:\n");

printf("|");

// Iterate over each process

for (int i = 0; i < n; ++i) {

// Execute the process

processes[i].completion\_time = current\_time + processes[i].burst\_time;

// Print Gantt chart

printf(" P%d |", processes[i].process\_id);

current\_time += processes[i].burst\_time;

}

printf("\n");

// Display completion time for each process

printf("\nProcess\t\tCompletion Time\n");

for (int i = 0; i < n; ++i) {

printf("P%d\t\t%d\n", processes[i].process\_id, processes[i].completion\_time);

}

}

int main() {

// Example usage

int n = 4; // Number of processes

struct Process processes[] = { {1, 0, 6, 3, 0}, {2, 1, 8, 1, 0}, {3, 2, 7, 4, 0}, {4, 3, 3, 2, 0} };

printf("Priority Scheduling:\n");

priorityScheduling(processes, n);

return 0;

}

* Output:

Priority Scheduling:

Gantt Chart:

| P3 | P1 | P4 | P2 |

Process Completion Time

P3 7

P1 13

P4 16

P2 24