**OPERATING SYSTEMS**

**CYCLE – I:**

1. **Write a program to simulate the following CPU Scheduling algorithms.**
2. **FCFS**

**Source Code:**

#include <stdio.h>

#include <stdlib.h>

void sort(int n, int p[100], int at[100], int bt[100]) {

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

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

if (at[i] > at[j]) {

int temp = at[i];

at[i] = at[j];

at[j] = temp;

temp = bt[i];

bt[i] = bt[j];

bt[j] = temp;

temp = p[i];

p[i] = p[j];

p[j] = temp;

}

}

}

}

int main() {

int n;

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

scanf("%d", &n);

int bt[100], at[100], tt[100], ct[100], wt[100], p[100];

printf("Enter the arrival time and burst time of processes:\n");

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

p[i] = i + 1;

scanf("%d %d", &at[i], &bt[i]);

}

sort(n, p, at, bt);

int sum = 0, sum1 = 0, sum2 = 0;

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

sum += bt[x];

ct[x] = sum;

}

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

tt[c] = ct[c] - at[c];

}

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

wt[g] = tt[g] - bt[g];

}

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

sum1 += tt[y];

sum2 += wt[y];

}

printf("Processes Arrival Time Burst Time Completion Time Turnaround Time Waiting Time\n");

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

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

p[x], at[x], bt[x], ct[x], tt[x], wt[x]);

}

printf("The average Turnaround Time is: %.2f\n", (float)sum1 / n);

printf("The average Waiting Time is: %.2f\n", (float)sum2 / n);

return 0;

}

**Expected input:**

Enter the number of processes: 3

Enter burst time for process 1: 24

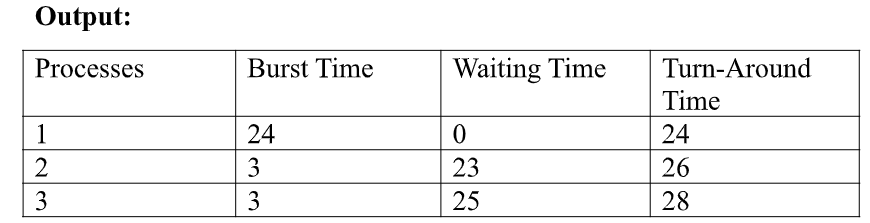
Enter arrival time for process 1: 0

Enter burst time for process 2: 3

Enter arrival time for process 2: 1

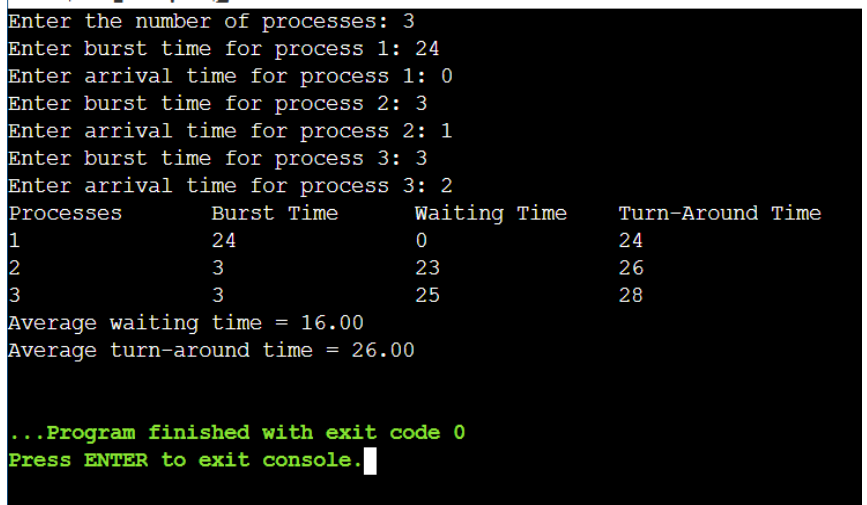
Enter burst time for process 3: 3

Enter arrival time for process 3: 2



Average waiting time = 16.00

Average turn-around time = 26.00

****

1. **SJF**

**Source Code:**

#include <stdio.h>

#include <stdlib.h>

void sort(int n, int p[100], int at[100], int bt[100]) {

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

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

if (bt[i] > bt[j]) {

int temp = at[i];

at[i] = at[j];

at[j] = temp;

temp = bt[i];

bt[i] = bt[j];

bt[j] = temp;

temp = p[i];

p[i] = p[j];

p[j] = temp;

}

}

}

}

int main() {

int n;

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

scanf("%d", &n);

int bt[100], at[100], tt[100], ct[100], wt[100], p[100];

printf("Enter the arrival time and burst time of processes:\n");

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

p[i] = i + 1;

scanf("%d %d", &at[i], &bt[i]);

}

sort(n, p, at, bt);

int sum = 0, sum1 = 0, sum2 = 0;

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

sum += bt[x];

ct[x] = sum;

}

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

tt[c] = ct[c] - at[c];

}

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

wt[g] = tt[g] - bt[g];

}

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

sum1 += tt[y];

sum2 += wt[y];

}

printf("Processes Arrival Time Burst Time Completion Time Turnaround Time Waiting Time\n");

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

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

p[x], at[x], bt[x], ct[x], tt[x], wt[x]);

}

printf("The average Turnaround Time is: %.2f\n", (float)sum1 / n);

printf("The average Waiting Time is: %.2f\n", (float)sum2 / n);

return 0;

}

**Expected input:**

Enter the number of processes: 3

Enter burst time for process 1: 6

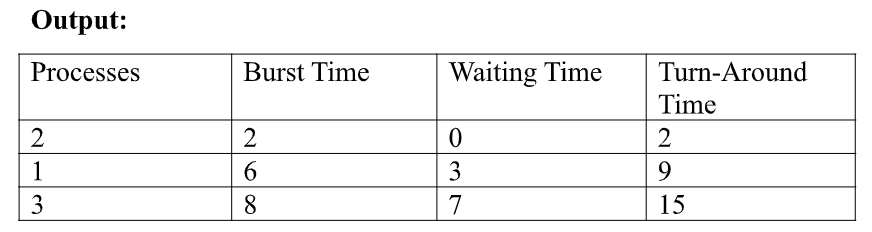
Enter arrival time for process 1: 1

Enter burst time for process 2: 2

Enter arrival time for process 2: 2

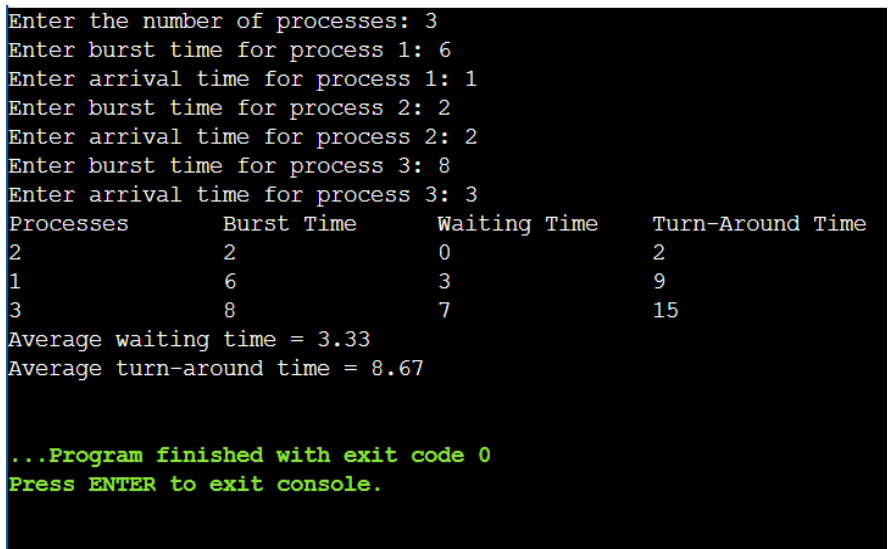
Enter burst time for process 3: 8

Enter arrival time for process 3: 3



Average waiting time = 3.33

Average turn-around time = 8.67



**c)Round Robin**

**Source Code**

#include <stdio.h>

void calculateWaitingTime(int n, int bt[], int wt[], int at[], int tq) {

int rem\_bt[n];

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

rem\_bt[i] = bt[i];

}

int t = 0;

while (1) {

int done = 1;

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

if (rem\_bt[i] > 0) {

done = 0;

if (rem\_bt[i] > tq) {

t += tq;

rem\_bt[i] -= tq;

} else {

t += rem\_bt[i];

wt[i] = t - bt[i] - at[i];

rem\_bt[i] = 0;

}

}

}

if (done == 1) {

break;

}

}

}

void calculateTurnAroundTime(int n, int bt[], int wt[], int tat[]) {

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

tat[i] = bt[i] + wt[i];

}

}

int main() {

int n, tq;

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

scanf("%d", &n);

int bt[n], at[n], wt[n], tat[n];

printf("Enter the time quantum: ");

scanf("%d", &tq);

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

printf("Enter burst time for process %d: ", i + 1);

scanf("%d", &bt[i]);

at[i] = 0;

}

calculateWaitingTime(n, bt, wt, at, tq);

calculateTurnAroundTime(n, bt, wt, tat);

printf("\nProcesses\tBurst Time\tWaiting Time\tTurn-Around Time\n");

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

printf("%d\t\t%d\t\t%d\t\t%d\n", i + 1, bt[i], wt[i], tat[i]);

}

float avg\_wt = 0, avg\_tat = 0;

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

avg\_wt += wt[i];

avg\_tat += tat[i];

}

avg\_wt /= n;

avg\_tat /= n;

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

printf("Average Turn-Around Time: %.2f\n", avg\_tat);

return 0;

}

**Expected input:**

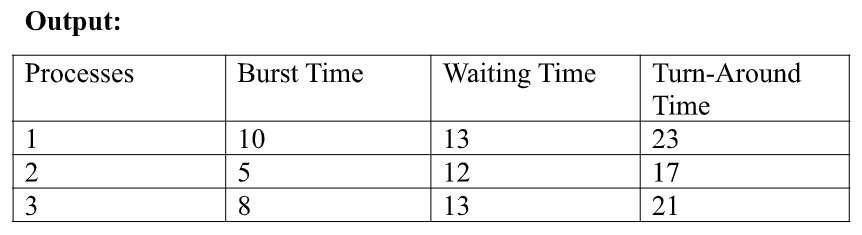
Enter the number of processes: 3

Enter the time quantum: 4

Enter burst time for process 1: 10

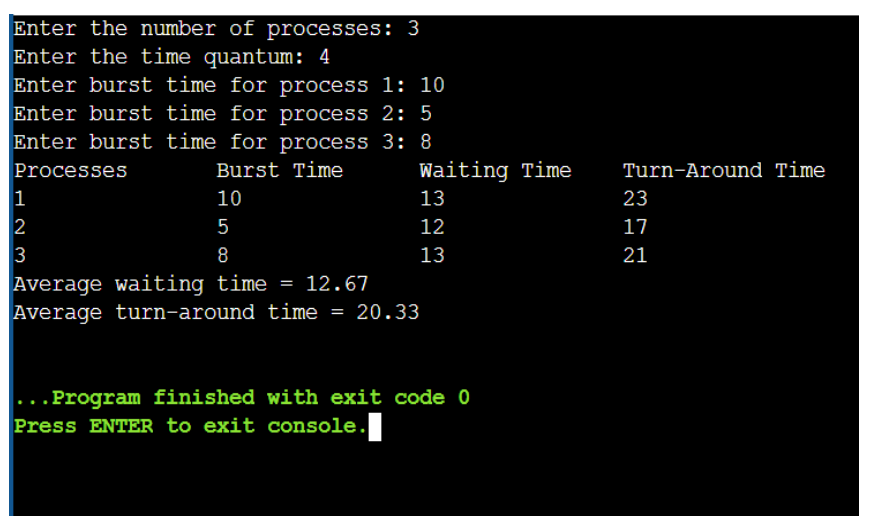
Enter burst time for process 2: 5

Enter burst time for process 3: 8



Average waiting time = 12.67

Average turn-around time = 20.33



**d)Priority**

**Source Code**

#include <stdio.h>

void priorityScheduling(int n, int burst\_time[], int priority[], int arrival\_time[], int waiting\_time[], int turnaround\_time[], int completion\_time[]) {

int i, j;

// Sort based on arrival time, then priority (Bubble Sort)

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

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

if (arrival\_time[j] > arrival\_time[j + 1] ||

(arrival\_time[j] == arrival\_time[j + 1] && priority[j] > priority[j + 1])) {

// Swap everything associated with the processes

int temp;

temp = burst\_time[j];

burst\_time[j] = burst\_time[j + 1];

burst\_time[j + 1] = temp;

temp = priority[j];

priority[j] = priority[j + 1];

priority[j + 1] = temp;

temp = arrival\_time[j];

arrival\_time[j] = arrival\_time[j + 1];

arrival\_time[j + 1] = temp;

}

}

}

// Calculation

int current\_time = 0;

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

if (current\_time < arrival\_time[i]) {

current\_time = arrival\_time[i];

}

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

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

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

current\_time = completion\_time[i]; // Update current time

}

}

int main() {

int num\_processes;

printf("Enter number of processes: ");

scanf("%d", &num\_processes);

int burst\_time[num\_processes], priority[num\_processes], arrival\_time[num\_processes];

int waiting\_time[num\_processes], turnaround\_time[num\_processes], completion\_time[num\_processes];

// Input

for (int i = 0; i < num\_processes; i++) {

printf("Enter Burst Time for P%d: ", i + 1);

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

printf("Enter Priority for P%d (lower is higher): ", i + 1);

scanf("%d", &priority[i]);

printf("Enter Arrival Time for P%d: ", i + 1);

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

}

// Call function to calculate scheduling

priorityScheduling(num\_processes, burst\_time, priority, arrival\_time, waiting\_time, turnaround\_time, completion\_time);

// Output

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

for (int i = 0; i < num\_processes; i++) {

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

i + 1, arrival\_time[i], burst\_time[i], priority[i],

waiting\_time[i], turnaround\_time[i], completion\_time[i]);

}

// Average Calculation and Output

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

for (int i = 0; i < num\_processes; i++) {

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

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

}

avg\_waiting /= num\_processes;

avg\_turnaround /= num\_processes;

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

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

return 0;

}

**Expected input:**

Enter the number of processes: 3

Enter burst time for process 1: 10

Enter arrival time for process 1: 0

Enter priority for process 1: 2

Enter burst time for process 2: 5

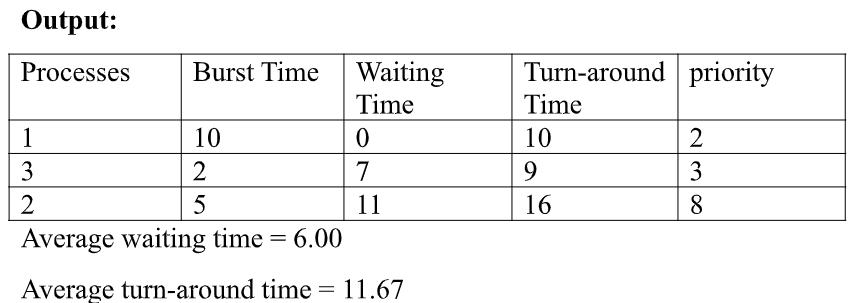
Enter arrival time for process 2: 1

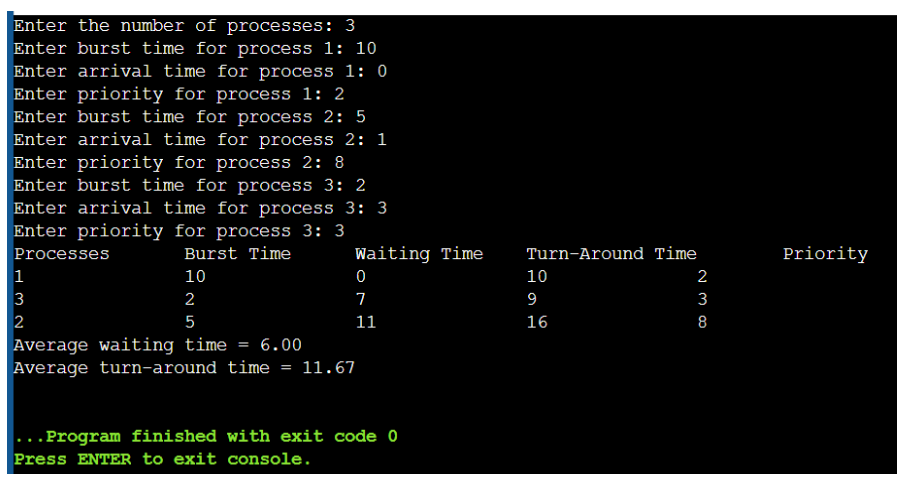
Enter priority for process 2: 1

Enter burst time for process 3: 8

Enter arrival time for process 3: 2

Enter priority for process 3: 3



****

**2.a) Write a program to implement Process management system calls viz., fork, exit, wait, waitpid, exec.**

**Source Code**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

int main() {

    pid\_t pid = fork();

    if (pid < 0) {

        perror("Fork failed");

        exit(1);

    }

    if (pid == 0) {

        char \*args[] = {"./fc", NULL};

        execvp(args[0], args);

        perror("exec failed");

        exit(1);

    } else {

        printf("Parent Process: PID = %d, Child PID = %d\n", getpid(), pid);

        int status;

        wait(&status);

        printf("Parent Process: Child has finished with status %d\n", status);

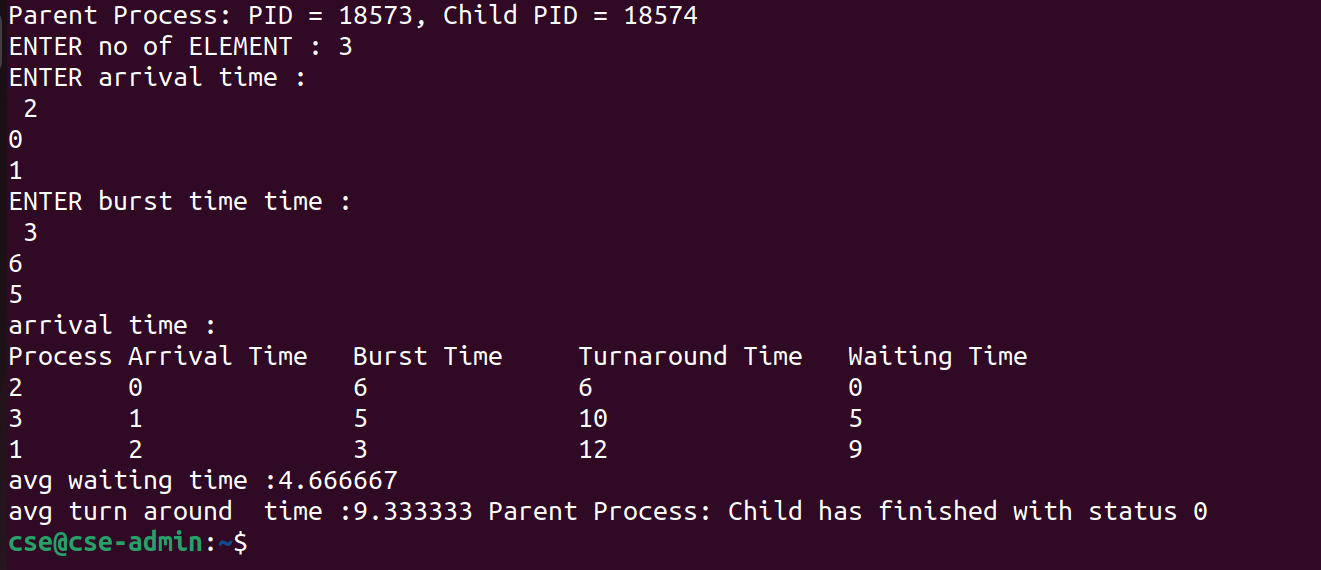
        exit(0);

    }

    return 0;

}

**OUTPUT**

****