#include <stdio.h>

#include <stdbool.h>

typedef struct

{

int pid;

int cpu\_time;

int io\_time;

int remaining\_cpu\_time;

int remaining\_io\_time;

int arrival\_time;

int turnaround\_time;

int waiting\_time;

bool is\_io;

} Process;

void roundRobinScheduling(Process processes[], int n, int time\_quantum)

{

int current\_time = 0;

int completed = 0;

int i;

while (completed != n)

{

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

{

if (processes[i].arrival\_time<= current\_time&& processes[i].remaining\_cpu\_time> 0)

{

if (processes[i].is\_io && processes[i].remaining\_io\_time> 0)

{

printf("Process %d is performing I/O at time %d.\n", processes[i].pid, current\_time);

if (processes[i].remaining\_io\_time<= time\_quantum)

{

current\_time += processes[i].remaining\_io\_time;

processes[i].remaining\_io\_time = 0;

processes[i].is\_io = false;

}

else

{

processes[i].remaining\_io\_time -= time\_quantum;

current\_time += time\_quantum;

}

}

else if (!processes[i].is\_io&& processes[i].remaining\_cpu\_time> 0)

{

printf("Process %d is executing on CPU at time %d.\n", processes[i].pid, current\_time);

if (processes[i].remaining\_cpu\_time<= time\_quantum)

{

current\_time += processes[i].remaining\_cpu\_time;

processes[i].remaining\_cpu\_time = 0;

processes[i].is\_io = true;

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

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

completed++;

}

else

{

processes[i].remaining\_cpu\_time -= time\_quantum;

current\_time += time\_quantum;

processes[i].is\_io = true;

}

}

}

}

}

}

int main()

{

int n, time\_quantum;

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

scanf("%d", &n);

Process processes[n];

printf("Enter CPU time for all process :");

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

{

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

}

printf("Enter I/O time for all process :");

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

{

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

}

printf("Enter arrival time for all process : ");

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

{

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

}

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

{

processes[i].pid = i + 1;

processes[i].remaining\_cpu\_time = processes[i].cpu\_time;

processes[i].remaining\_io\_time = processes[i].io\_time;

processes[i].is\_io = false;

}

printf("Enter the time quantum: ");

scanf("%d", &time\_quantum);

roundRobinScheduling(processes, n, time\_quantum);

printf("\nProcess\tTurnaround Time\tWaiting Time\n");

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

{

printf("%d\t%d\t\t%d\n", processes[i].pid, processes[i].turnaround\_time, processes[i].waiting\_time);

}

return 0;

}

OUTPUT

Enter the number of processes: 2

Enter CPU time for all process :2 2

Enter I/O time for all process :2 2

Enter arrival time for all process : 0 0

Enter the time quantum: 2

Process 1 is executing on CPU at time 0.

Process 2 is executing on CPU at time 2.

Process Turnaround Time Waiting Time

1 2 0

2 4 2