#include <stdio.h>

#include <stdlib.h>

#define MAX 10

void print\_gantt\_chart(int p[], int n);

int p[MAX],n,

bt[MAX],wt[MAX],tat[MAX],r[50];

int k=0,quantum=0;

// Function to calculate turn around time

int turnarroundtime(int p[], int n,

int bt[], int wt[], int tat[]) {

// calculating turnaround time by adding

// bt[i] + wt[i]

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

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

return 1;

}

// Function to find the waiting time for all

// processes

int waitingtime(int p[], int n,

int bt[], int wt[], int quantum) {

// Make a copy of burst times bt[] to store remaining

// burst times.

int rem\_bt[n];

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

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

int t = 0; // Current time

// Keep traversing processes in round robin manner

// until all of them are not done.

while (1) {

int done = 1;

// Traverse all processes one by one repeatedly

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

// If burst time of a process is greater than 0

// then only need to process further

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

done = 0; // There is a pending process

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

// Increase the value of t i.e. shows

// how much time a process has been processed

r[k]=p[i];

k++;

t += quantum;

// Decrease the burst\_time of current process

// by quantum

rem\_bt[i] -= quantum;

}

// If burst time is smaller than or equal to

// quantum. Last cycle for this process

else {

// Increase the value of t i.e. shows

// how much time a process has been processed

t = t + rem\_bt[i];

// Waiting time is current time minus time

// used by this process

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

// As the process gets fully executed

// make its remaining burst time = 0

rem\_bt[i] = 0;

}

}

}

// If all processes are done

if (done == 1)

break;

}

return 1;

}

// Function to calculate average time

int findavgTime(int p[], int n, int bt[],

int quantum) {

int wt[n], tat[n], total\_wt = 0, total\_tat = 0;

// Function to find waiting time of all processes

waitingtime(p, n, bt, wt, quantum);

// Function to find turn around time for all processes

turnarroundtime(p, n, bt, wt, tat);

// Display processes along with all details

int i;

printf("+-----+------------+--------------+-----------------+\n");

printf("| PID | Burst Time | Waiting Time | Turnaround Time |\n");

printf("+-----+------------+--------------+-----------------+\n");

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

printf("| %d | %d | %d | %d |\n"

, p[i], bt[i], wt[i], tat[i]);

printf("+-----+------------+--------------+-----------------+\n");

}

print\_gantt\_chart(p,n);

printf("\n");

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

total\_wt = total\_wt + wt[i];

total\_tat = total\_tat + tat[i];

}

printf("Average waiting time = %f", (float)total\_wt / (float)n);

printf("\nAverage turnaround time = %f\n", (float)total\_tat / (float)n);

return 1;

}

void print\_gantt\_chart(int p[], int n)

{

int i, j;

// print top bar

printf(" ");

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

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

printf(" ");

}

printf("\n|");

// printing process id in the middle

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

for(j=0; j<quantum; j++) printf(" ");

printf("P%d", r[i]);

for(j=0; j<quantum; j++) printf(" ");

printf("|");

}

printf("\n ");

// printing bottom bar

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

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

printf(" ");

}

printf("\n");

// printing the time line

int h=0;

printf("0");

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

if(h<11)

{

for(j=0; j<=quantum; j++) printf(" ");

h=h+quantum;

printf("%d",h);

}

else{

for(j=0; j<=quantum-1; j++) printf(" ");

h=h+quantum;

printf("%d",h);

}

}

}

// main function

int main() {

int i, j;

printf("Enter total number of process:");

scanf("%d", &n);

printf("Enter burst time for each process:\n");

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

p[i]= i;

printf("P[%d] : ", i);

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

wt[i]=tat[i]=0;

}

printf("Enter the quantum size");

scanf("%d",&quantum);

int quantum = 2;

findavgTime(p, n, bt, quantum);

return 0;

}

