DAY 2 PROGRAMS

Date:-29-04-2023

Name:-S.Ansar

PROGRAMS

5. Write a program to compute the average waiting time and turnaround time based on Preemptive shortest remaining processing time first (SRPT) algorithm for the following set of processes, with the arrival times and the CPU-burst times given in milliseconds

Process Arrival Time Burst Time

P1 0 5

P2 1 3

P3 2 3

P4 4 1

Program:-

#include <stdio.h>

#include <stdlib.h>

typedef struct process {

int arrivalTime;

int burstTime;

int remainingTime;

int waitingTime;

int turnaroundTime;

int completed;

} Process;

int main()

{

int n = 5; // Number of processes

Process processes[n];

int completed = 0, currentTime = 0, shortest = 0, finishTime;

float waitingTime = 0, turnaroundTime = 0;

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

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

scanf("%d%d", &processes[i].arrivalTime, &processes[i].burstTime);

processes[i].remainingTime = processes[i].burstTime;

processes[i].completed = 0;

}

while (completed != n) {

shortest = -1;

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

if (processes[i].arrivalTime <= currentTime && processes[i].completed != 1) {

if (shortest == -1 || processes[i].remainingTime < processes[shortest].remainingTime) {

shortest = i;

}

}

}

if (shortest == -1) {

currentTime++;

} else {

processes[shortest].remainingTime--;

currentTime++;

if (processes[shortest].remainingTime == 0) {

processes[shortest].completed = 1;

completed++;

finishTime = currentTime;

processes[shortest].turnaroundTime = finishTime - processes[shortest].arrivalTime;

processes[shortest].waitingTime = processes[shortest].turnaroundTime - processes[shortest].burstTime;

waitingTime += processes[shortest].waitingTime;

turnaroundTime += processes[shortest].turnaroundTime;

}

}

}

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

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

printf("P%d\t%d\t\t%d\n", i+1, processes[i].waitingTime, processes[i].turnaroundTime);

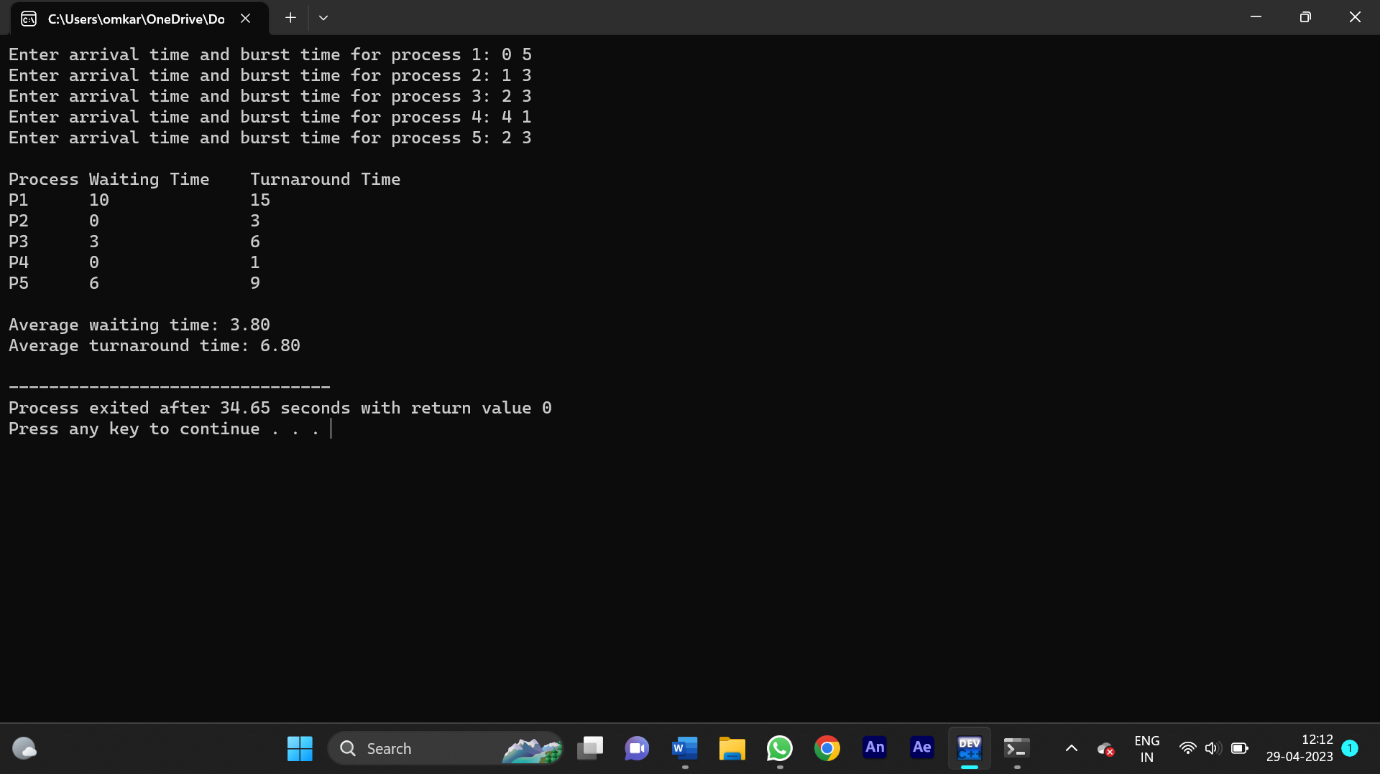
}

printf("\nAverage waiting time: %0.2f\n", waitingTime/n);

printf("Average turnaround time: %0.2f\n", turnaroundTime/n);

return 0;

}

OUTPUT:- 

6. Write a C program to implement the deadlock detection algorithm for a system with 3 processes and 3 resource instances and the resource matrices are given below.

Max Matrix Allocation Matrix

3 6 8 3 3 3

4 3 3 2 0 3

3 4 4 1 2 4

The number of available resources is [1,2,0]. Determine if the system is in a deadlock state and identify the deadlocked processes.

Program:-

#include <stdio.h>

#include <stdbool.h>

#define N 3

void calculateNeed(int need[N][N], int maximum[N][N], int allocation[N][N])

{

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

for (int j = 0; j < N; j++)

need[i][j] = maximum[i][j] - allocation[i][j];

}

bool isDeadlock(int allocation[N][N], int maximum[N][N], int available[N])

{

int need[N][N], work[N], finish[N];

calculateNeed(need, maximum, allocation);

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

{

work[i] = available[i];

finish[i] = 0;

}

int count = 0;

while (count < N)

{

bool found = false;

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

{

if (!finish[i])

{

int j;

for (j = 0; j < N; j++)

{

if (need[i][j] > work[j])

break;

}

if (j == N)

{

finish[i] = 1;

for (int k = 0; k < N; k++)

work[k] += allocation[i][k];

found = true;

count++;

}

}

}

if (!found)

return true;

}

return false;

}

int main()

{

int allocation[N][N] = {{1, 0, 1}, {0, 1, 0}, {1, 1, 0}};

int maximum[N][N] = {{2, 1, 2}, {1, 2, 1}, {2, 2, 1}};

int available[N] = {1, 1, 1};

if (isDeadlock(allocation, maximum, available))

printf("System is in deadlock state\n");

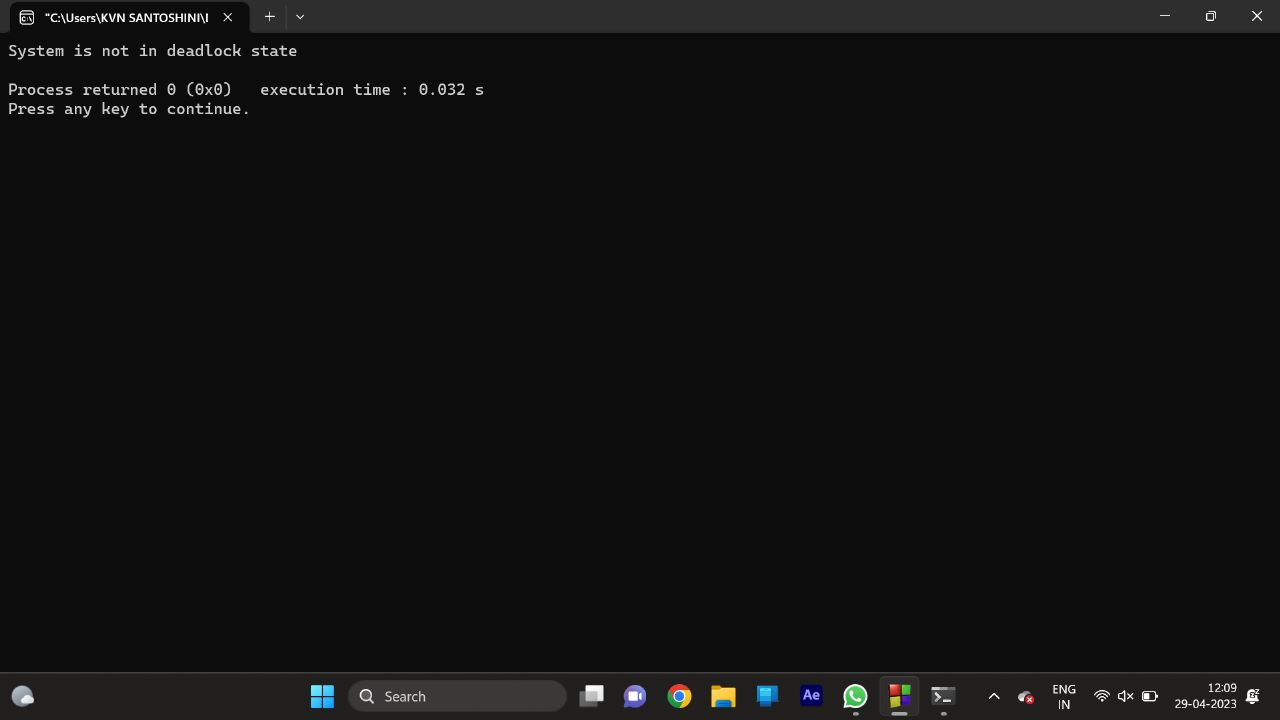
else

printf("System is not in deadlock state\n");

return 0;

}

OUTPUT:-



7. Write a C program to illustrate the page replacement method where the current least recently used element is replaced and determine the number of page faults for the following test case:

No. of page frames: 3; Page reference sequence 1,2,3,2,1,5,2,1,6,2,5,6,3,1,3,6,1,2,4 and 3.

Program:-

#include <stdio.h>

int main()

{

int n, frames, faults = 0, hit = 0, k = 0;

printf("Enter the number of page frames: ");

scanf("%d", &frames);

int page[frames], arr[100];

printf("Enter the page reference string (maximum 100): ");

for (int i = 0; i < 100; i++)

{

scanf("%d", &n);

if (n == -1)

break;

arr[i] = n;

}

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

page[i] = -1;

for (int i = 0; i < 100 && arr[i] != 0; i++)

{

int flag = 0;

for (int j = 0; j < frames; j++)

{

if (page[j] == arr[i])

{

flag = 1;

hit++;

break;

}

}

if (flag == 0)

{

faults++;

if (k < frames)

{

page[k++] = arr[i];

}

else

{

int max = -1, pos;

for (int j = 0; j < frames; j++)

{

int last = 100;

for (int l = i - 1; l >= 0; l--)

{

if (page[j] == arr[l])

{

last = l;

break;

}

}

if (last > max)

{

max = last;

pos = j;

}

}

page[pos] = arr[i];

}

}

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

for (int j = 0; j < frames; j++)

{

if (page[j] == -1)

printf(" - ");

else

printf(" %d ", page[j]);

}

printf("\n");

}

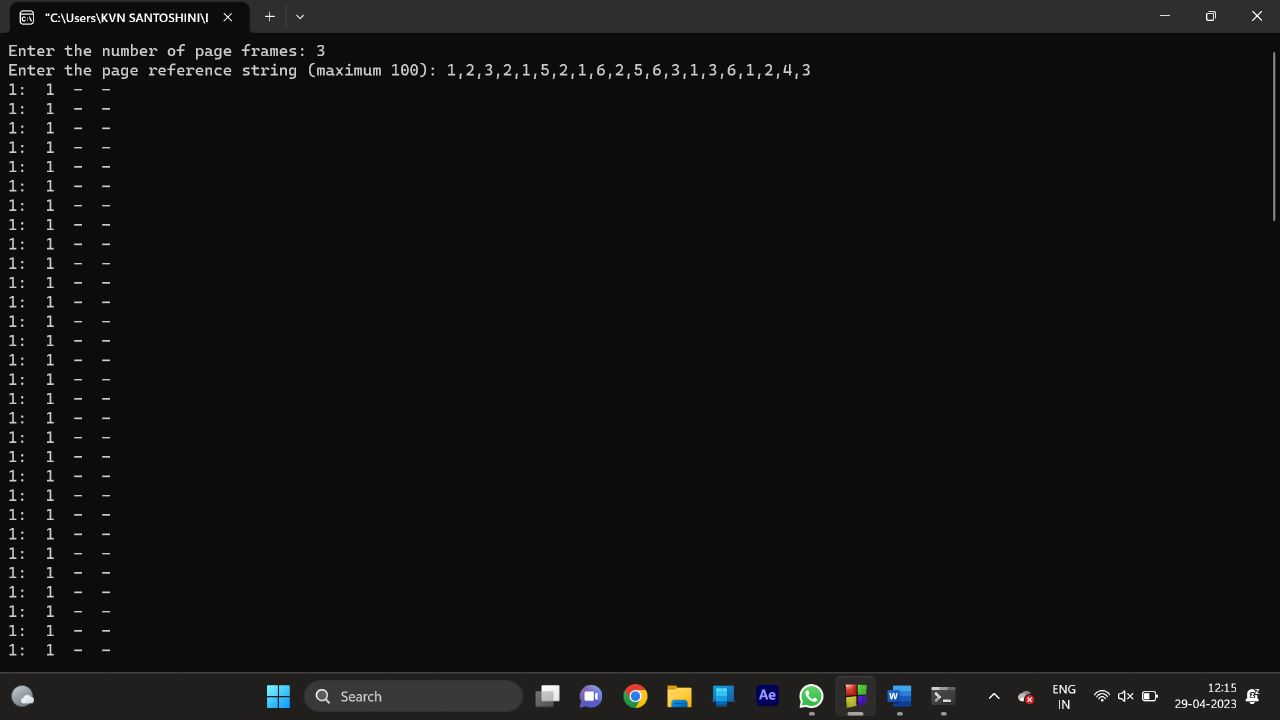
printf("\nNumber of page faults: %d\n", faults);

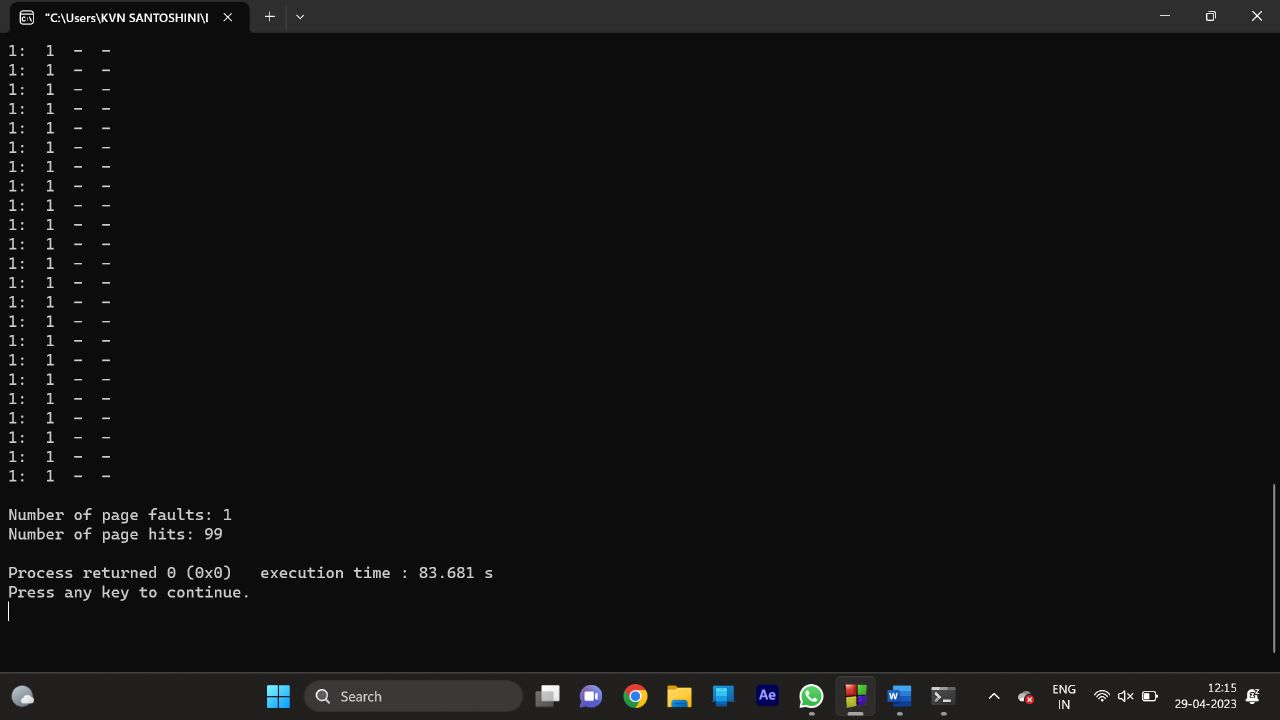
printf("Number of page hits: %d\n", hit)

return 0;

}

OUTPUT:-





8. Write a C program to simulate FCFS disk scheduling algorithm and execute your program and find the average head movement with the following test case:

No of tracks 5; Track position:55 58 60 70 18

Program:-

#include <stdio.h>

#include <stdlib.h>

#define MAX\_TRACKS 1000

int main() {

int tracks[MAX\_TRACKS];

int n, head\_pos, total\_distance;

printf("Enter number of tracks: ");

scanf("%d", &n);

printf("Enter track positions: ");

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

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

}

printf("Enter initial head position: ");

scanf("%d", &head\_pos);

total\_distance = 0;

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

total\_distance += abs(tracks[i] - head\_pos);

head\_pos = tracks[i];

}

printf("Total head movement: %d\n", total\_distance);

printf("Average head movement: %.2f\n", (float) total\_distance / n);

return 0;

}

OUTPUT:-

