**fcfs**  
#include <stdio.h>

struct P {

int id, at, bt, wt, tat;

};

void calcTimes(struct P p[], int n) {

int totalWT = 0, totalTAT = 0;

p[0].wt = 0;

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

p[i].wt = p[i - 1].wt + p[i - 1].bt;

}

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

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

}

printf("\nP\tAT\tBT\tWT\tTAT\n");

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

printf("%d\t%d\t%d\t%d\t%d\n", p[i].id, p[i].at, p[i].bt, p[i].wt, p[i].tat);

totalWT += p[i].wt;

totalTAT += p[i].tat;

}

printf("\nAvg WT: %.2f\n", (float)totalWT / n);

printf("Avg TAT: %.2f\n", (float)totalTAT / n);

}

int main() {

int n;

printf("Enter number of processes: ");

scanf("%d", &n);

struct P p[n];

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

p[i].id = i + 1;

printf("Enter AT and BT for P%d: ", p[i].id);

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

}

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

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

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

struct P temp = p[i];

p[i] = p[j];

p[j] = temp;

}

}

}

calcTimes(p, n);

return 0;

}

**sjfs**

#include <stdio.h>

struct P {

int id, at, bt, wt, tat;

};

void calcTimes(struct P p[], int n) {

int totalWT = 0, totalTAT = 0;

p[0].wt = 0;

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

p[i].wt = p[i - 1].wt + p[i - 1].bt;

}

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

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

}

printf("\nP\tAT\tBT\tWT\tTAT\n");

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

printf("%d\t%d\t%d\t%d\t%d\n", p[i].id, p[i].at, p[i].bt, p[i].wt, p[i].tat);

totalWT += p[i].wt;

totalTAT += p[i].tat;

}

printf("\nAvg WT: %.2f\n", (float)totalWT / n);

printf("Avg TAT: %.2f\n", (float)totalTAT / n);

}

int main() {

int n;

printf("Enter number of processes: ");

scanf("%d", &n);

struct P p[n];

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

p[i].id = i + 1;

printf("Enter AT and BT for P%d: ", p[i].id);

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

}

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

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

if (p[i].bt > p[j].bt || (p[i].bt == p[j].bt && p[i].at > p[j].at)) {

struct P temp = p[i];

p[i] = p[j];

p[j] = temp;

}

}

}

calcTimes(p, n);

return 0;

}

**Round robain**

#include <stdio.h>

struct P {

int id, bt, rem\_bt, wt, tat;

};

void calcTimes(struct P p[], int n, int quantum) {

int totalWT = 0, totalTAT = 0, t = 0,completed = 0;

while (completed < n) {

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

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

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

t += quantum;

p[i].rem\_bt -= quantum;

} else {

t += p[i].rem\_bt;

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

p[i].tat = t;

p[i].rem\_bt = 0;

completed++;

}

}

}

}

printf("\nP\tBT\tWT\tTAT\n");

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

printf("%d\t%d\t%d\t%d\n", p[i].id, p[i].bt, p[i].wt, p[i].tat);

totalWT += p[i].wt;

totalTAT += p[i].tat;

}

printf("\nAvg WT: %.2f\n", (float)totalWT / n);

printf("Avg TAT: %.2f\n", (float)totalTAT / n);

}

int main() {

int n, quantum;

printf("Enter number of processes: ");

scanf("%d", &n);

struct P p[n];

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

p[i].id = i + 1;

printf("Enter burst time for P%d: ", p[i].id);

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

p[i].rem\_bt = p[i].bt; // Remaining burst time

}

printf("Enter time quantum: ");

scanf("%d", &quantum);

calcTimes(p, n, quantum);

return 0;

}

**Fifo**#include <stdio.h>

void fifoPageReplacement(int pages[], int n, int frames[], int capacity) {

int index = 0, pageFaults = 0;

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

int found = 0;

for (int j = 0; j < capacity; j++) {

if (frames[j] == pages[i]) {

found = 1;

break;

}

}

if (!found) {

frames[index] = pages[i];

index = (index + 1) % capacity;

pageFaults++;

}

printf("Page %d -> ", pages[i]);

for (int j = 0; j < capacity; j++) {

if (frames[j] != -1)

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

else

printf("- ");

}

printf("\n");

}

printf("\nTotal Page Faults: %d\n", pageFaults);

}

int main() {

int n, capacity;

printf("Enter number of pages: ");

scanf("%d", &n);

int pages[n];

printf("Enter the pages: ");

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

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

}

printf("Enter number of frames: ");

scanf("%d", &capacity);

int frames[capacity];

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

frames[i] = -1;

}

fifoPageReplacement(pages, n, frames, capacity);

return 0;

}

**LRU**#include <stdio.h>

int findLRU(int used[], int capacity) {

int min = used[0], pos = 0;

for (int i = 1; i < capacity; i++) {

if (used[i] < min) {

min = used[i];

pos = i;

}

}

return pos;

}

void lruPageReplacement(int pages[], int n, int frames[], int capacity) {

int used[capacity], pageFaults = 0, time = 0;

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

frames[i] = -1;

used[i] = 0;

}

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

int found = 0;

for (int j = 0; j < capacity; j++) {

if (frames[j] == pages[i]) {

found = 1;

used[j] = ++time;

break;

}

}

if (!found) {

int lruIndex = findLRU(used, capacity);

frames[lruIndex] = pages[i];

used[lruIndex] = ++time;

pageFaults++;

}

printf("Page %d -> ", pages[i]);

for (int j = 0; j < capacity; j++) {

if (frames[j] != -1)

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

else

printf("- ");

}

printf("\n");

}

printf("\nTotal Page Faults: %d\n", pageFaults);

}

int main() {

int n, capacity;

printf("Enter number of pages: ");

scanf("%d", &n);

int pages[n];

printf("Enter the pages: ");

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

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

}

printf("Enter number of frames: ");

scanf("%d", &capacity);

int frames[capacity];

lruPageReplacement(pages, n, frames, capacity);

return 0;

}

**Write a Shell Script to accept a number and find Even or ODD**

**ans:**

**echo "Enter a number:"**

**read num**

**if [ $((num % 2)) -eq 0 ]; then**

**echo "$num is Even"**

**else**

**echo "$num is Odd"**

**fi**

**Write a Shell Script to find the Factorial of a given number.**

**ans:**

**echo "Enter a number:"**

**read num**

**fact=1**

**for ((i = 1; i <= num; i++))**

**do**

**fact=$((fact \* i))**

**done**

**echo "Factorial of $num is $fact"**

**Write a Shell Script to find the Greatest of the given three numbers.**

**ans:**

**echo "Enter three numbers:"**

**read a b c**

**if [ $a -gt $b ] && [ $a -gt $c ]; then**

**echo "$a is the greatest"**

**elif [ $b -gt $a ] && [ $b -gt $c ]; then**

**echo "$b is the greatest"**

**else**

**echo "$c is the greatest"**

**fi**

**Write a Shell Script to accept numbers and print sorted numbers.**

**ans:**

**echo "Enter numbers separated by spaces:"**

**read -a numbers**

**sorted=($(for i in "${numbers[@]}"; do echo $i; done | sort -n))**

**echo "Sorted numbers: ${sorted[@]}"**