**// Q6: Write a C program to implement FCFS.**

#include <stdlib.h>

#include <stdio.h>

int main()

{

int n, bt[20], wt[20], tat[20], i, j;

float avwt = 0, avtat = 0;

printf("Enter total number of processes(maximum 20):");

scanf("%d", &n);

printf("\nEnter Process Burst Time\n");

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

{

printf("P[%d]:", i + 1);

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

}

wt[0] = 0;

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

{

wt[i] = 0;

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

wt[i] += bt[j];

}

printf("\nProcess\t\tBurst Time\tWaiting Time\tTurnaround Time");

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

{

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

avwt += wt[i];

avtat += tat[i];

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

}

avwt /= i;

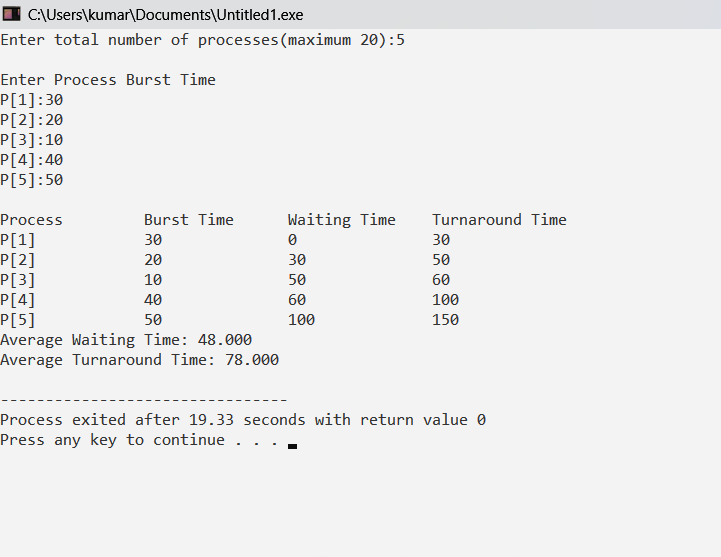
avtat /= i;

printf("\nAverage Waiting Time: %.3f", avwt);

printf("\nAverage Turnaround Time: %.3f\n", avtat);

return 0;

}

****

**// Q7 – a : Write a C program to implement SJF – Non Preemptive..**

#include <stdlib.h>

#include <stdio.h>

int main()

{

int bt[20], p[20], wt[20], tat[20], i, j, n, total = 0, pos, temp;

float avg\_wt, avg\_tat;

printf("Enter number of process : ");

scanf("%d", &n);

printf("\nEnter Burst Time : \n");

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

{

printf("P%d : ", i + 1);

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

p[i] = i + 1;

}

// sorting of burst times

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

{

pos = i;

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

{

if (bt[j] < bt[pos])

pos = j;

}

temp = bt[i];

bt[i] = bt[pos];

bt[pos] = temp;

temp = p[i];

p[i] = p[pos];

p[pos] = temp;

}

wt[0] = 0;

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

{

wt[i] = 0;

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

wt[i] += bt[j];

total += wt[i];

}

avg\_wt = (float)total / n;

total = 0;

printf("\nProcess\t Burst Time \tWaiting Time\tTurnaround Time");

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

{

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

total += tat[i];

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

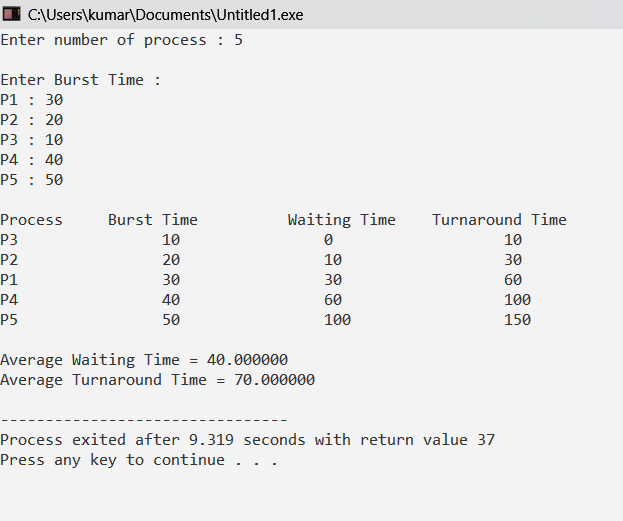
}

avg\_tat = (float)total / n;

printf("\n\nAverage Waiting Time = %f", avg\_wt);

printf("\nAverage Turnaround Time = %f\n", avg\_tat);

}

****

**// Q7 – b : Write a C program to implement SJF – Preemptive..**

#include <stdio.h>

#include <stdlib.h>

int main()

{

int arrival\_time[10], burst\_time[10], temp[10];

int i, smallest, count = 0, time, limit;

double wait\_time = 0, turnaround\_time = 0, end;

float average\_waiting\_time, average\_turnaround\_time;

printf("\nEnter the Total Number of Processes : \t");

scanf("%d", &limit);

printf("\nEnter Details of %d Processes\n", limit);

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

{

printf("\nEnter Arrival Time : \t");

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

printf("Enter Burst Time : \t");

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

temp[i] = burst\_time[i];

}

burst\_time[9] = 9999;

for (time = 0; count != limit; time++)

{

smallest = 9;

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

{

if (arrival\_time[i] <= time && burst\_time[i] < burst\_time[smallest] && burst\_time[i] > 0)

{

smallest = i;

}

}

burst\_time[smallest]--;

if (burst\_time[smallest] == 0)

{

count++;

end = time + 1;

wait\_time = wait\_time + end - arrival\_time[smallest] - temp[smallest];

turnaround\_time = turnaround\_time + end - arrival\_time[smallest];

}

}

average\_waiting\_time = wait\_time / limit;

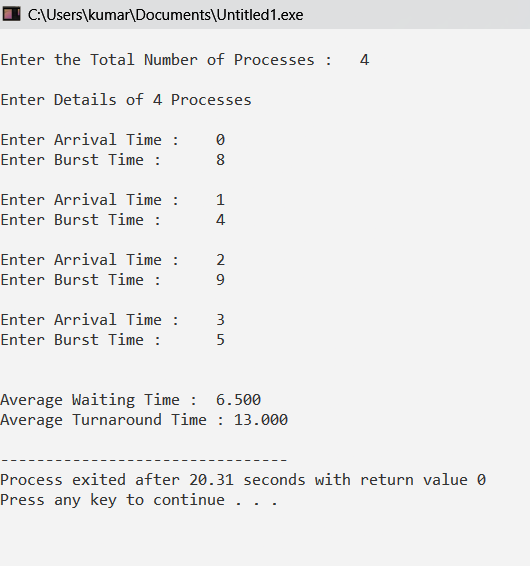
average\_turnaround\_time = turnaround\_time / limit;

printf("\n\nAverage Waiting Time : \t%.3lf\n", average\_waiting\_time);

printf("Average Turnaround Time : %.3lf\n", average\_turnaround\_time);

return 0;

}



**// Q8: Write a C program to implement Priority Scheduling.**

#include <stdio.h>

int main()

{

int x, n, p[10], pp[10], pt[10], w[10], t[10], awt, atat, i,j;

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

scanf("%d", &n);

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

{

printf("\nFor process no %d Burst Time and Priority : ", i + 1);

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

p[i] = i + 1;

}

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

{

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

{

if (pp[i] < pp[j])

{

x = pp[i];

pp[i] = pp[j];

pp[j] = x;

x = pt[i];

pt[i] = pt[j];

pt[j] = x;

x = p[i];

p[i] = p[j];

p[j] = x;

}

}

}

w[0] = 0;

awt = 0;

t[0] = pt[0];

atat = t[0];

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

{

w[i] = t[i - 1];

awt += w[i];

t[i] = w[i] + pt[i];

atat += t[i];

}

printf("\n\nJob \t Burst Time \t Wait Time \t Turn Around Time Priority \n");

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

printf("\n %d \t\t %d \t\t %d \t\t %d \t\t %d \n", p[i], pt[i], w[i], t[i], pp[i]);

awt /= n;

atat /= n;

printf("\nAverage Wait Time : %d \n", awt);

printf("\nAverage Turn Around Time : %d \n", atat);

}

****

**// Q9: Write a C program to implement FIFO Page Replacement Policy.**

#include <stdio.h>

int main()

{

int frames, t,i;

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

scanf("%d", &frames);

printf("Enter the size of incoming stream : ");

scanf("%d", &t);

int incomingStream[t];

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

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

int pageFaults = 0;

int m, n, s, pages;

pages = sizeof(incomingStream)/sizeof(incomingStream[0]);

printf("Incoming \t Frame 1 \t Frame 2 \t Frame 3");

int temp[frames];

for(m = 0; m < frames; m++)

{

temp[m] = -1;

}

for(m = 0; m < pages; m++)

{

s = 0;

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

{

if(incomingStream[m] == temp[n])

{

s++;

pageFaults--;

}

}

pageFaults++;

if((pageFaults <= frames) && (s == 0))

{

temp[m] = incomingStream[m];

}

else if(s == 0)

{

temp[(pageFaults - 1) % frames] = incomingStream[m];

}

printf("\n");

printf("%d\t\t",incomingStream[m]);

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

{

if(temp[n] != -1)

printf(" %d\t\t", temp[n]);

else

printf(" - \t\t");

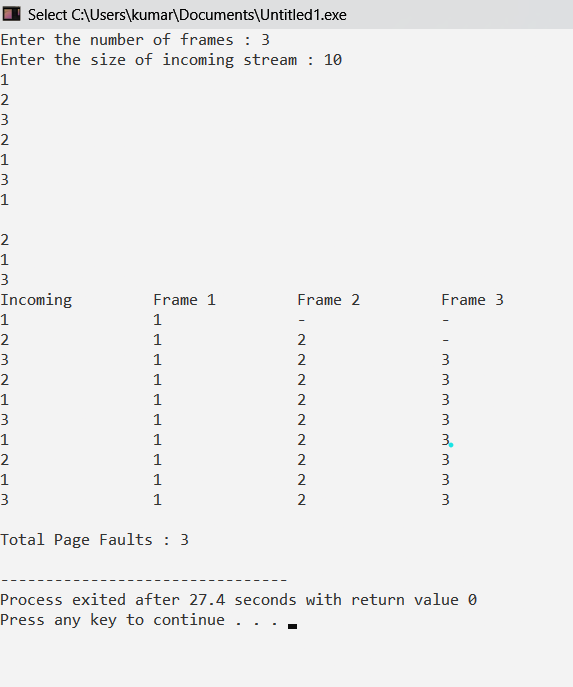
}

}

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

return 0;

}



**// Q10: Write a C program to implement LRU PRP.**

#include <stdio.h>

int findLRU(int time[], int n)

{

int i, minimum = time[0], pos = 0;

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

{

if (time[i] < minimum)

{

minimum = time[i];

pos = i;

}

}

return pos;

}

int main()

{

int no\_of\_frames, no\_of\_pages, frames[10], pages[30], counter = 0, time[10], flag1, flag2, i, j, pos, faults = 0;

printf("Enter number of frames: ");

scanf("%d", &no\_of\_frames);

printf("Enter size of the incoming stream : ");

scanf("%d", &no\_of\_pages);

printf("Enter the stream : ");

for (i = 0; i < no\_of\_pages; ++i)

{

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

}

for (i = 0; i < no\_of\_frames; ++i)

{

frames[i] = -1;

}

for (i = 0; i < no\_of\_pages; ++i)

{

flag1 = flag2 = 0;

for (j = 0; j < no\_of\_frames; ++j)

{

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

{

counter++;

time[j] = counter;

flag1 = flag2 = 1;

break;

}

}

if (flag1 == 0)

{

for (j = 0; j < no\_of\_frames; ++j)

{

if (frames[j] == -1)

{

counter++;

faults++;

frames[j] = pages[i];

time[j] = counter;

flag2 = 1;

break;

}

}

}

if (flag2 == 0)

{

pos = findLRU(time, no\_of\_frames);

counter++;

faults++;

frames[pos] = pages[i];

time[pos] = counter;

}

printf("\n");

for (j = 0; j < no\_of\_frames; ++j)

{

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

}

}

printf("\n\nTotal Page Faults = %d", faults);

return 0;

}

