/\*

# PROGRAM TO IMPLEMENT SJF SCHEDULING ALGORITHM

Ashis Solomon

CS4B 17

MDL20CS035

\*/

**CODE:**

#include <stdio.h>

int n, i, j, yes, temp, completionTime[10], proc[10], loop, smallest, current\_time, sorted[10], arrival[10] = {0}, visited[10] = {0}, tat[10], order[10], wt[10], burst[10] = {0}, visited[10];

float total\_tat = 0, total\_wt = 0;

void swap(int \*xp, int \*yp)

{

int temp = \*xp;

\*xp = \*yp;

\*yp = temp;

}

void Sort(int arr[], int n)

{

int i, j;

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

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

if (arr[i] > arr[j])

{

swap(&arr[i], &arr[j]);

swap(&proc[i], &proc[j]);

swap(&arrival[i], &arrival[j]);

}

}

int main()

{

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

printf("\n SHORTEST JOB FIRST \n\n");

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

printf("Enter no of process : ");

scanf("%d", &n);

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

{

printf("\nArrival time of process %d is :", i);

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

printf("\nBurst time of process %d is :", i);

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

proc[i] = i;

visited[i] = 0;

}

smallest = arrival[1];

j = smallest;

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

{

if (smallest > arrival[loop])

{

smallest = arrival[loop];

j = loop;

}

}

current\_time = smallest;

j = 1;

Sort(burst, n);

while (j <= n)

{

yes = 0;

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

{

if (visited[i] == 0 && current\_time >= arrival[i])

{

yes = 1;

order[j] = proc[i];

visited[i] = 1;

wt[i] = current\_time - arrival[i];

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

completionTime[i] = current\_time + burst[i];

total\_wt += wt[i];

total\_tat += tat[i];

current\_time = completionTime[i];

++j;

break;

}

}

if (yes == 0)

{

++current\_time;

}

}

total\_wt = total\_wt / n;

total\_tat = total\_tat / n;

printf("\norder in which process get executed :\t");

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

{

printf("%d \t ", order[i]);

}

printf("\n\n arrival\_time Burst\_time Turn\_around\_time waiting\_time completion time");

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

{

printf("\nprocess%d %d %d %d %d %d", proc[i], arrival[i], burst[i], tat[i], wt[i], completionTime[i]);

}

printf("\n\n Average waiting time is %.2f", total\_wt);

printf("\n Average turn around time is %.2f", total\_tat);

}

**OUTPUT:**

