7. Construct a C program to implement a non-preemptive SJF algorithm.

#include<iostream>

using namespace std;

void swap(int \*a, int \*b) {

int temp = \*a;

\*a = \*b;

\*b = temp;

}

void arrangeArrival(int num, int mat[][3]) {

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

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

if(mat[1][j] > mat[1][j+1]) {

for(int k=0; k<5; k++) {

swap(mat[k][j], mat[k][j+1]);

}

}

}

}

}

void completionTime(int num, int mat[][3]) {

int temp, val;

mat[3][0] = mat[1][0] + mat[2][0];

mat[5][0] = mat[3][0] - mat[1][0];

mat[4][0] = mat[5][0] - mat[2][0];

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

temp = mat[3][i-1];

int low = mat[2][i];

for(int j=i; j<num; j++) {

if(temp >= mat[1][j] && low >= mat[2][j]) {

low = mat[2][j];

val = j;

}

}

mat[3][val] = temp + mat[2][val];

mat[5][val] = mat[3][val] - mat[1][val];

mat[4][val] = mat[5][val] - mat[2][val];

for(int k=0; k<6; k++) {

swap(mat[k][val], mat[k][i]);

}

}

}

int main() {

int num = 3, temp;

int mat[6][3] = {1, 2, 3, 3, 6, 4, 2, 3, 4};

cout<<"Before Arrange...\n";

cout<<"Process ID\tArrival Time\tBurst Time\n";

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

cout<<mat[0][i]<<"\t\t"<<mat[1][i]<<"\t\t"<<mat[2][i]<<"\n";

}

arrangeArrival(num, mat);

completionTime(num, mat);

cout<<"Final Result...\n";

cout<<"Process ID\tArrival Time\tBurst Time\tWaiting Time\tTurnaround Time\n";

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

cout<<mat[0][i]<<"\t\t"<<mat[1][i]<<"\t\t"<<mat[2][i]<<"\t\t"<<mat[4][i]<<"\t\t"<<mat[

5][i]<<"\n";

}

}

Output:

