2.Design and implement a scheduler using a preemptive SJF algorithm.

Shortest Job First (Non Preemptive) - algorithm

1. BEGIN
2. Get number of processes (numberOfProcesses) as input from user.
3. P[i]=array of processes.
4. while(input){
5. get burst time for each process
6. }
7. while(total burst time){
8. sort\_increasing\_order(P) //according to arrival time (first) and then remaining burst //time
9. total burst time—
10. }
11. Calculate Waiting time , average waiting time and average turn around time for each process in P.
12. Display Gantt Chart.
13. END

/\*

\* SJFP.cpp

\*

\* Created on: 29-Aug-2012

\* Author: Lokesh Chandra Basu

\* Enrol. no: 10114026

\* Branch: CSE III yr

\*

\*/

**#include**<iostream>

**using** **namespace** std;

**struct** process{

**int** pId ;

**int** burstTime ;

};

**int** **main**(){

**int** numberOfProcesses;

//number of process

cout << "\nNUMBER OF PROCESS: ";

cin >> numberOfProcesses;

**struct** process p[numberOfProcesses];

**struct** process temp;

**int** i;

**int** j;

**int** waitingTime = 0;

**float** totalWaitingTime = 0;

//burst time

**for**(i=0; i<numberOfProcesses; i++){

p[i].pId = i+1 ;

cout << "================================\n";

cout << "PROCESS " << i+1 << "\n";

cout << "\tBURST TIME: ";

cin >> p[i].burstTime;

totalWaitingTime += p[i].burstTime;

}

//sort

**for**(i=0; i<numberOfProcesses; i++){

**for**(j=0; j<numberOfProcesses; j++){

**if**(p[j].burstTime>p[i].burstTime){

temp = p[i];

p[i] = p[j] ;

p[j] = temp;

}

}

}

//display

cout <"\n========================================================\n";

cout << " PROCESS\t BURST TIME\t WAITING TIME\n\n";

**for**(i=0;i<numberOfProcesses;i++){

cout <<"\tP" << p[i].pId << "\t\t" << p[i].burstTime;

cout << "\t\t" << waitingTime << "\n";

waitingTime = waitingTime + p[i].burstTime ;

}

cout << "========================================================\n";

//average waiting time

cout << "AVERAGE WAITING TIME = ";

cout << totalWaitingTime/numberOfProcesses << "\n\n" ;

**return** 0;

}

**OUTPUT:**

enter the number of processes: 3

enter the proc information:

pid at bt

1

0

5

3

1

2

2

2

1

the proc information:

pid at bt

1 0 5

3 1 2

2 2 1

gantt chart:0 p[1]1 p[3]2 p[3]3 p[2]4 p[1]5 p[1]6 p[1]7 p[1]8

average wt=1.333333, average tat=4.000000

the proc information:

pid at bt wt tat

1 0 5 3 8

3 1 2 0 2

2 2 1 1 2