NATIONAL INSTITUTE OF TECHNOLOGY MIZORAM

OS LAB ASSIGNMENT

Name: NIRAJ KUMAR

Department: CSE

Enrollment No. : BT19CS031

1. Implement the Shortest Job First CPU scheduling algorithm. Read inputs from a file (format given below).

Inputs:

The program will read from an input file containing a list of processes along with other data require for

scheduling.

The input file will look like this:

P1 0 20.0

P2 2 15.0

P3 6 27.0

p4 4 36.0

• The file containing the information on the processes will have each process on a separate line. The

processes will be in the file in the order in which they arrive at the OS.

• Each line will have a process name that will be a string.

• Following the name will be the arrival time of the process

• Following arrival time will be the total burst time.

Outputs:

The program must print out the time taken by each process to complete (turnaround time) and the wait

time and compute the average turnaround time for all processes.

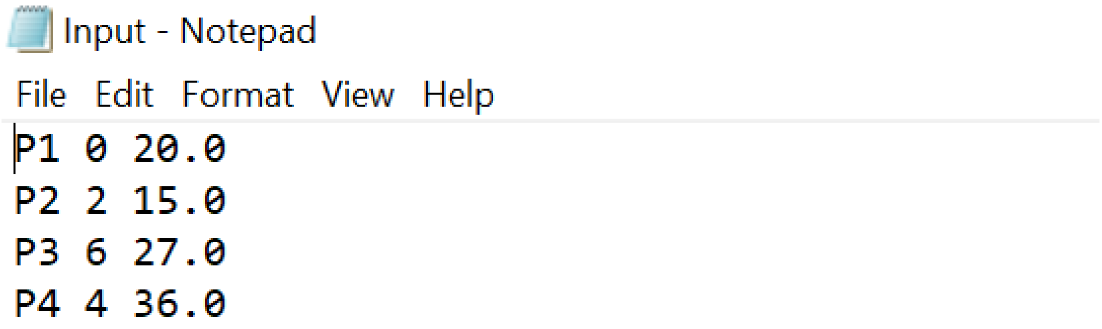
```
// CPU Scheduling, SJF Non-Preemptive


#include<stdio.h>
#include<stdlib.h>
#include<iostream>
#include<string>
#include <cstdlib>
#include <fstream>
#include <sstream>
#include <algorithm>
#include <iomanip>
#include<vector>
using namespace std;

// Process class containing info about the process
class Process
{
private:
string name; //Name of process
int process_id;
int processed_status;
int arrival_time;
int burst_time;
int completion_time;
int turn_around_time;
int wait_time;

public:

//constructor to initialise member variables
Process(string name,int arrival_time,int burst_time,int
process_id)
{
this->name = name;
this->arrival_time = arrival_time;
this->burst_time = burst_time;
this->process_id = process_id;
this->processed_status = 0;
this->completion_time = 0;
this->turn_around_time = 0;
this->wait_time = 0;
```

```
}

int getBurstTime()
{
    return burst_time;
}

int getArrivalTime()
{
    return arrival_time;
}
int getProcessID()
{
    return process_id;
}

int getTurnAroundTime()
{
    return turn_around_time;
}

int getWaitTime()
{
    return wait_time;
}

void setTurnAroundTime()
{
 turn_around_time = completion_time - arrival_time;
}

void setWaitTime()
{
 wait_time = turn_around_time - burst_time;
}

void displayDetails()
{
cout<<name<<"\t"<<arrival_time<<"\t\t"<<burst_time<<"\t\t"

<<completion_time<<"\t\t"<<turn_around_time<<"\t\t"<<
        wait_time<<endl;
```

```
}

friend void calculateCompletionTime(Process **);

};

//Comparator for sort function

bool compare(Process *p1, Process *p2)
{
 if(p1->getBurstTime()!= p2->getBurstTime())
    return p1->getBurstTime() < p2->getBurstTime();

 else if(p1->getArrivalTime()!= p2->getArrivalTime())
    return p1->getArrivalTime()< p2->getArrivalTime();

 else
    return p1->getProcessID()< p2->getProcessID();
}


void calculateCompletionTime(Process **p)
{
 int time_counter=0;
//vector container in cpp stl

 vector <Process*> readyqueue;
 for(int count=1;count<= 4;)
 {
 for (int i=0;i<= 3;i++)
 if(p[i]->arrival_time <= time_counter && p[i]-
>processed_status == 0)
 readyqueue.push_back(p[i]);

 if(!readyqueue.empty())
 {
 stable_sort(readyqueue.begin(),readyqueue.end(),compare);
 time_counter=time_counter + readyqueue[0]->burst_time;
 readyqueue[0]->processed_status = 1;
 readyqueue[0]->completion_time = time_counter;
 readyqueue[0]->setTurnAroundTime();
 readyqueue[0]->setWaitTime();
 readyqueue.clear();
 count++;
 }
 else
 time_counter++;
}
}


int main()
{

// Array to store arrival and burst time for all the
process
int at[4],bt[4];
string lines[4];

//File Handling , taking input from file
ifstream fio;
fio.open("C:\\Users\\NIRAJ
KUMAR\\Desktop\\c++\\Input.txt");
for (int i=0;i<=3;i++)
{
getline(fio,lines[i]);
stringstream gk1(lines[i].substr(3,1));
gk1 >> at[i];
stringstream gk2(lines[i].substr(5,4));
gk2 >> bt[i];
}
fio.close();

// Creating objects of Process class
//And initialising them

Process **p=(Process**)malloc(sizeof(Process*)*4);
p[0]=new Process("P1",at[0],bt[0],1);
p[1]=new Process("P2",at[1],bt[1],2);
p[2]=new Process("P3",at[2],bt[2],3);
p[3]=new Process("P4",at[3],bt[3],4);


calculateCompletionTime(p);
```

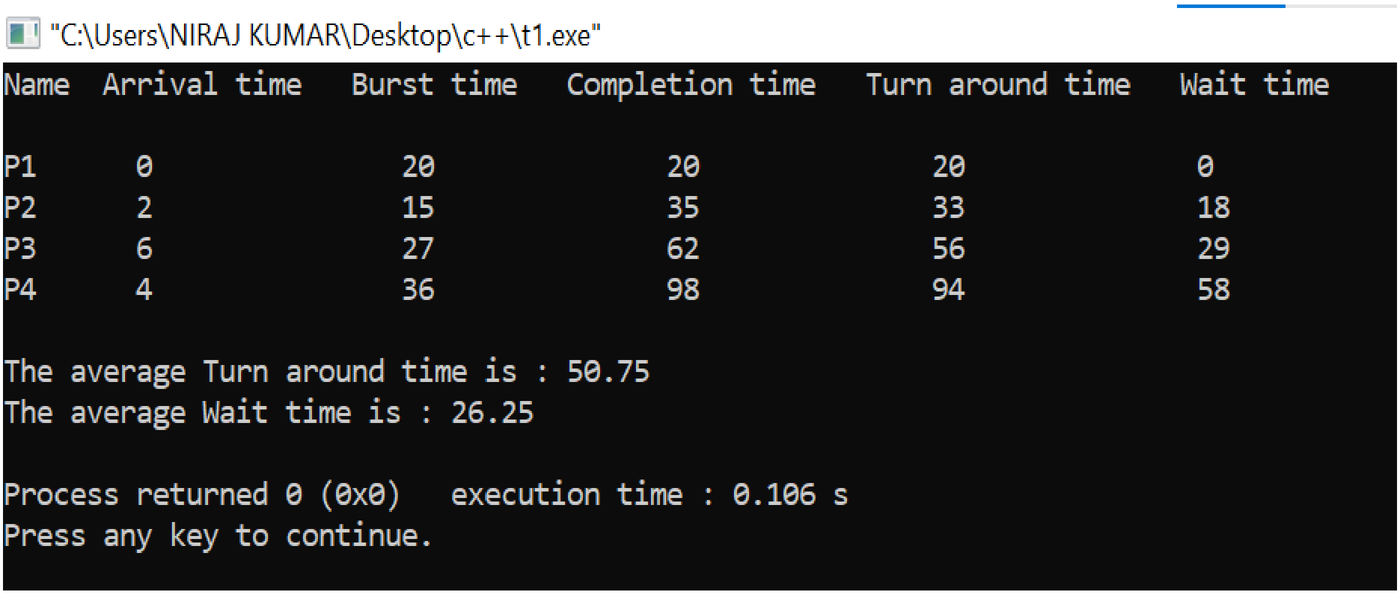
```
//Displaying Details

cout<<"Name  Arrival time   Burst time   Completion time
Turn around time   Wait time\n\n";

for (int i=0;i<=3;i++)
 p[i]->displayDetails();

double total_wait_time = 0,total_turn_around_time = 0;
for (int i=0;i<=3;i++)
{
total_turn_around_time += p[i]->getTurnAroundTime();
total_wait_time += p[i]->getWaitTime();
}
cout<<"\nThe average Turn around time is :
"<<total_turn_around_time/4<<endl;
cout<<"The average Wait time is :
"<<total_wait_time/4<<endl;

return 0;


}
```

INPUT FILE :



```
P1 0 20.0
P2 2 15.0
P3 6 27.0
P4 4 36.0
```

OUTPUT :



"C:\Users\NIRAJ KUMAR\Desktop\c++\t1.exe"

```
Name  Arrival time   Burst time   Completion time   Turn around time   Wait time

P1      0               20              20              20              0
P2      2               15              35              33              18
P3      6               27              62              56              29
P4      4               36              98              94              58

The average Turn around time is : 50.75
The average Wait time is : 26.25

Process returned 0 (0x0)   execution time : 0.106 s
Press any key to continue.
```