```
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
using namespace std;

struct Process{
string processNumber;
int arrivalTime;
int cpuTime;
int totalTime;
int waitTime;
int endTime;
int startTime;
int turnAroundTime;
};

class Queue {
private:
int head;
int tail;
int count;
int maxSize;
vector<Process> processer;

public:
int getSize(){ //returns size
return processer.size();
```

```
}

bool isEmpty(){ //checks if is empty
if(processer.size() == 0){
return true;
}
return false;
}

void enqueue(Process p){ //enqueue
processer.push_back(p);
}

Process dequeue(){ //dequeue
Process temp = processer.front();
processer.erase(processer.begin());
return temp;
}

vector<Process> getProcesses(){ //gets procesesser
return processer;
}

void setProcesses(vector<Process> p){
processer = p;
}
```

```
};

void fcfs(vector<Process> p){
int time = 0;
int totalWaitTime=0;
double procAmount = (double) p.size();
double averageTime;

for(int i = 0; i < p.size(); i++){
p.at(i).startTime = time;
time = time + p.at(i).cpuTime;
p.at(i).endTime = time;
p.at(i).turnAroundTime=p.at(i).endTime-p.at(i).arrivalTime; //turn around time

p.at(i).waitTime=p.at(i).turnAroundTime-p.at(i).cpuTime; //wait time

totalWaitTime = totalWaitTime + p.at(i).waitTime; //total wait time
}

averageTime = totalWaitTime/procAmount;

cout<<endl;
//Displays to console
cout << "FCFS (non-preemptive): " << endl;
//Ghantt Chart
```

```
for(int i = 0; i < p.size(); i++){
if(i == 0){
cout << "Time | " << p.at(i).arrivalTime << " | " << p.at(i).endTime;
}
else if(i == p.size()-1){
cout << " | " << p.at(i).endTime << " |";
}
else {
cout << " | " << p.at(i).endTime;
}
}
cout << endl;


for(int i = 0; i < p.size(); i++){
if(i == 0){
cout << "CPU | " << p.at(i).processNumber;
}
else if(i == p.size()-1){
cout << " | " << p.at(i).processNumber << " | ";
}
else {
cout << " | " << p.at(i).processNumber;
}


}
cout<<endl;
for(int i = 0; i < p.size(); i++){
cout<<"-----------------------------------------"<<endl;
cout << "Process: " << p.at(i).processNumber << endl;
```

```
cout << "Arrival Time: " << p.at(i).arrivalTime << endl;
cout << "Service Time: " << p.at(i).cpuTime << endl;
cout << "Start Time: " << p.at(i).startTime << endl;
cout << "Finished at: " << p.at(i).endTime << endl;
cout << "Turn Around Time: "<< p.at(i).turnAroundTime<<endl;
cout << "Wait Time: "<<p.at(i).waitTime<<endl;
}
cout<<"-----------------------------------------"<<endl;


cout<<"Total time required is: "<< time<<" time units"<<endl;
cout<<"Average waiting time is: "<<averageTime<<" time units"<<endl;
cout<<endl;
}


int main(){
Queue processes;
fstream myFile; // Creates file, which lets you read and write from file
string line;
Process p;
int i = 0;
myFile.open("jobs.txt"); //opens jobs.txt file, which has a list of animes
if(myFile.fail()){ // if file doesn't open, user will know by getting a :(
cout << "Not good. :(" << endl;
exit(1);
}
while(!myFile.eof()){
myFile >> p.processNumber >> p.arrivalTime >> p.cpuTime;
p.startTime = 0;
p.totalTime = 0;
```

```
p.endTime = 0;
processes.enqueue(p);
i++;
}
myFile.close();


fcfs(processes.getProcesses());
return 0;
}
```

Output

```
FCFS (non-preemptive):
Time | 0 | 3 | 9 | 13 | 18 | 20 |
CPU | A | B | C | D | E |
-------------------------------------------------
Process: A
Arrival Time: 0
Service Time: 3
Start Time: 0
Finished at: 3
Turn Around Time: 3
Wait Time: 0
-------------------------------------------------
Process: B
Arrival Time: 2
Service Time: 6
Start Time: 3
Finished at: 9
Turn Around Time: 7
Wait Time: 1
-------------------------------------------------
Process: C
Arrival Time: 4
Service Time: 4
Start Time: 9
Finished at: 13
Turn Around Time: 9
Wait Time: 5
-------------------------------------------------
Process: D
Arrival Time: 6
Service Time: 5
Start Time: 13
Finished at: 18
Turn Around Time: 12
Wait Time: 7
-------------------------------------------------
Process: E
Arrival Time: 8
Service Time: 2
Start Time: 18
Finished at: 20
Turn Around Time: 12
Wait Time: 10
-------------------------------------------------
Total time required is: 20 time units
Average waiting time is: 4.6 time units
```

jobs.txt

A 0 3

B 2 6

C 4 4

D 6 5

E 8 2