Operating System

Scheduling

Program: communication and electronics.

Section: 8

Name	ID
Mohamed Nasser Mohamed Ibrahim	18011632
Mohamed Badr Saad Zaghlool Mohamed	18015047
Mohammed Adel Omar Bayoumy	18011514
Seif Mohamed Ashraf	18015038
Ziad Mahmmoud Fawzy	18010745

Code:

Main.cpp

```
#include <iostream>
#include "Timeline.hpp"
#include "firstServed.hpp"
#include "printStats.hpp"
#include "prompt.hpp"
#include "readFile.hpp"
#include "roundRobin.hpp"
int main() {
Choice choice = prompt();
  int x;
  std::cout<< "enter number of testcase:";
  std::cin >> x;
  std::vector<Process> processVec;
  switch(x) {
  case 1:
    {processVec = readFile("testcase1.txt");
    }
    break;
  case 2:
    {processVec = readFile("testcase2.txt");
   }
    break;
  default:
    {processVec = readFile("testcase3.txt");
    }
    break;}
Timeline stats;
  switch (choice.mode) {
    case 0:
      stats = firstServed(processVec);
      printStats("outputFCFS.txt", stats);
      break;
    case 1:
      stats = roundRobin(processVec, choice.quanta);
      printStats("outputRR.txt", stats);
      break;
    default:
      break;
  }
}
```

Prompt.hpp

```cpp
#ifndef PROMPT
#define PROMPT

#include <iostream>

struct Choice {
  int mode, quanta = 0;
};

Choice prompt() {
  Choice choice;

  std::cout << "Mode of operation:\n"
            << "0: First Come First Served (FCFS)\n"
            << "1: Round Robin\n";

  std::cin >> choice.mode;

  switch (choice.mode) {
    case 0:
      std::cout << "You chose FCFS.\n";
      break;
    case 1:
      std::cout << "You chose round robin.\n";
      std::cout << "Please enter the time quanta for each process.\n";
      std::cin >> choice.quanta;
      while (std::cin.fail() || !choice.quanta) {
        std::cin.clear();
        std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
        std::cout << "Sorry, invalid input.\n"
                  << "Please, enter an integer.\n";
        std::cin >> choice.quanta;
      }
      break;
    default:
      std::cout << "Invalid input, please try again!\n";
      choice = prompt();
      break;
  }

  return choice;
}
#endif
```

readFile.hpp

```
#ifndef READFILE
#define READFILE

#include <fstream>
#include <iostream>
#include <sstream>
#include <vector>

#include "Process.hpp"

std::vector<Process> readFile(std::string fileName) {
  std::vector<Process> processVec;
  std::ifstream inputFile(fileName);
  std::vector<std::vector<std::string>> lineVec;

  if (inputFile.is_open()) {
    std::string line;
    while (getline(inputFile, line)) {
      std::vector<std::string> tempProcessVec;
      std::stringstream stream(line);
      std::string token;
      while (stream >> token) {
        tempProcessVec.push_back(token);
      }
      lineVec.push_back(tempProcessVec);
    }
    inputFile.close();
  }

  for (auto i: lineVec) {
    if (i.size() == 4) {
      Process process( std::stoi(i[0]),std::stoi(i[1]), std::stoi(i[2]),
std::stoi(i[3]));
      processVec.push_back(process);

    } else {
      std::cout << "***Invalid input***\n";
    }
  }

  return processVec;
}
#endif
```

Timeline.hpp

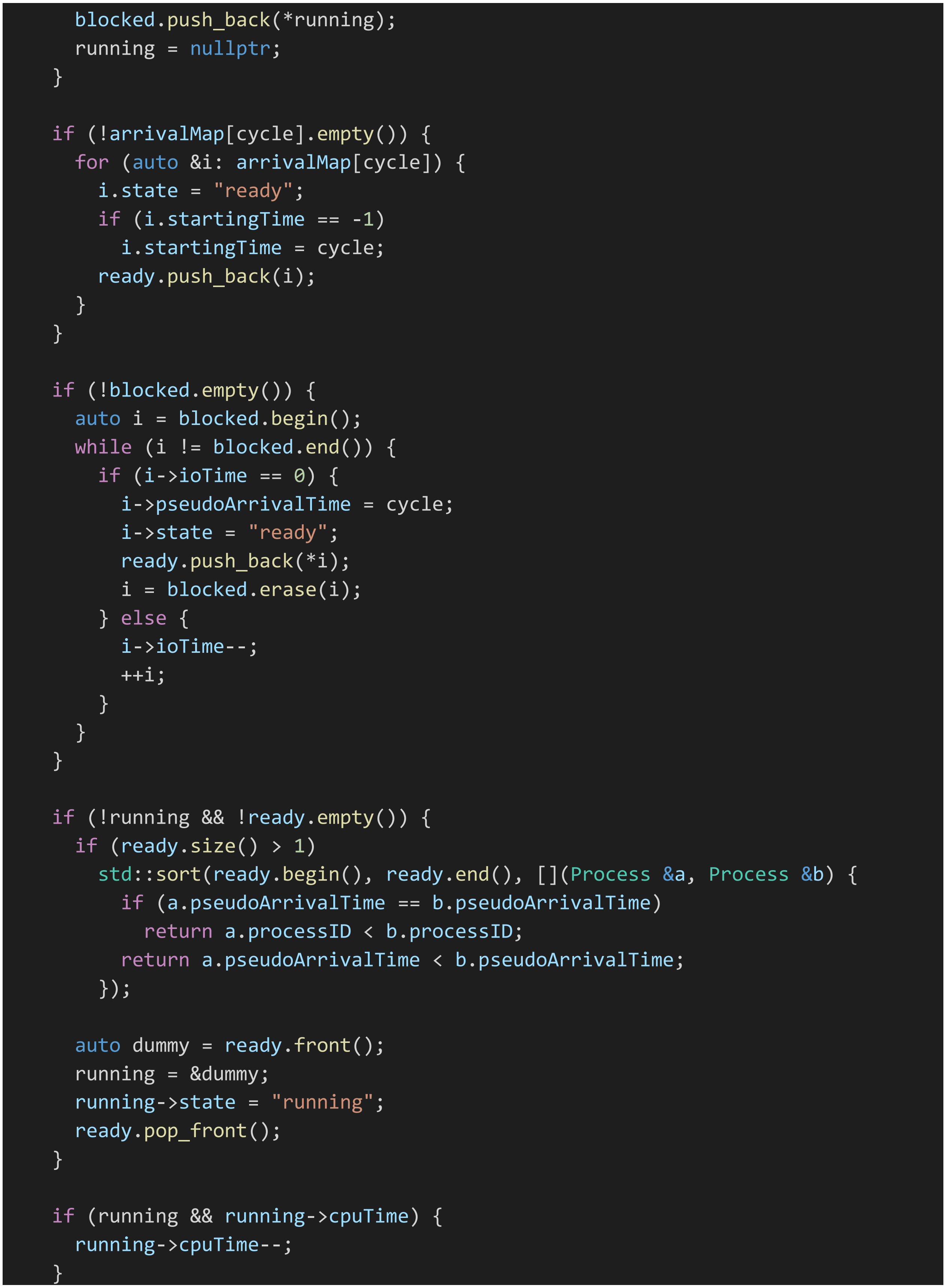
```
#ifndef TIMELINE
#define TIMELINE

#include <vector>

#include "Process.hpp"

struct Timeline {
  std::vector<std::vector<Process>> state;
  std::vector<Process> processes;
};

#endif
```

Process.cpp

```
#ifndef PROCESSSTRUCT
#define PROCESSSTRUCT

#include <iostream>

struct Process {
  int processID = -1;
  int cpuTime, ioTime, arrivalTime, pseudoArrivalTime, quanta = -1;
  int turnaroundTime, finishingTime, startingTime = -1;
  std::string state = "waiting";

  Process(int processID, int cpuTime, int ioTime, int arrivalTime)
      : processID(processID),
        cpuTime(cpuTime),
        ioTime(ioTime),
        arrivalTime(arrivalTime),
        pseudoArrivalTime(arrivalTime){};
  bool operator==(const struct Process &a) const {
    return (a.processID == this->processID);
  };
  bool operator!=(const struct Process &a) const {
    return (a.processID != this->processID);
  };
};
#endif
```

FirstServed.hpp

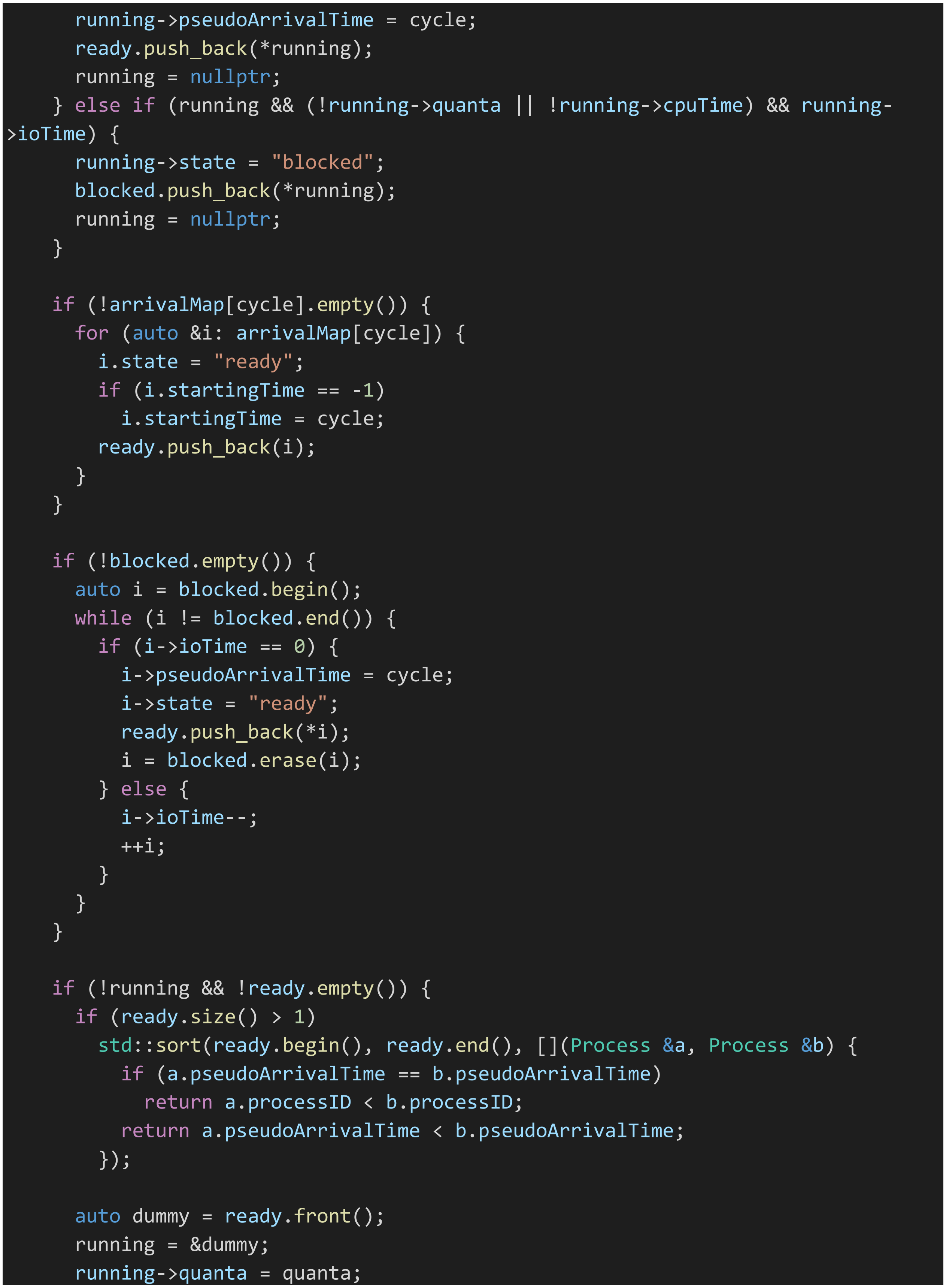
```
#ifndef FCFS
#define FCFS

#include <algorithm>
#include <deque>
#include <iostream>
#include <list>
#include <map>
#include <vector>

#include "Process.hpp"
#include "Timeline.hpp"

Timeline firstServed(std::vector<Process> &processes) {
  Timeline returnStruct;
  int cycle = 0;

  std::sort(processes.begin(), processes.end(), [](auto &a, auto &b) {
    return a.arrivalTime < b.arrivalTime;
  });

  Process *running = nullptr;
  std::list<Process> blocked;
  std::deque<Process> ready;
  std::map<int, std::vector<Process>> arrivalMap;

  for (auto &i: processes) {
    arrivalMap[i.arrivalTime].push_back(i);
  }

  while (running || !blocked.empty() || !ready.empty() || arrivalMap.rbegin()-
>first > cycle) {
    std::vector<Process> temp;

    if (running && !running->cpuTime && !running->ioTime) {
      running->finishingTime = cycle - 1;
      running->turnaroundTime = running->finishingTime - running->startingTime +
1;
      running->state = "terminated";
      returnStruct.processes.push_back(*running);
      running = nullptr;
    } else if (running && !running->cpuTime && running->ioTime) {
      running->state = "blocked";
```



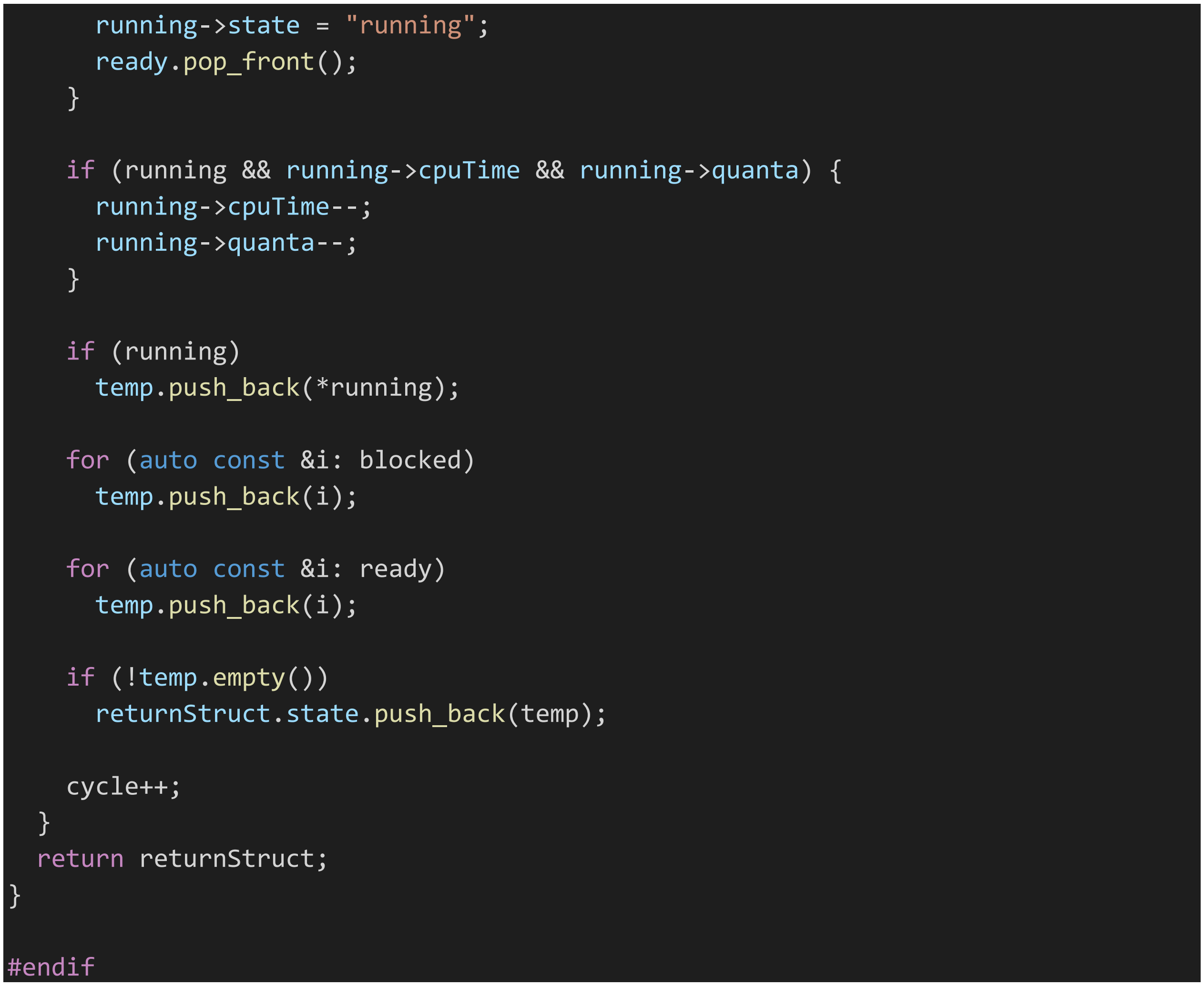
```
      blocked.push_back(*running);
      running = nullptr;
    }

    if (!arrivalMap[cycle].empty()) {
      for (auto &i: arrivalMap[cycle]) {
        i.state = "ready";
        if (i.startingTime == -1)
          i.startingTime = cycle;
        ready.push_back(i);
      }
    }

    if (!blocked.empty()) {
      auto i = blocked.begin();
      while (i != blocked.end()) {
        if (i->ioTime == 0) {
          i->pseudoArrivalTime = cycle;
          i->state = "ready";
          ready.push_back(*i);
          i = blocked.erase(i);
        } else {
          i->ioTime--;
          ++i;
        }
      }
    }

    if (!running && !ready.empty()) {
      if (ready.size() > 1)
        std::sort(ready.begin(), ready.end(), [](Process &a, Process &b) {
          if (a.pseudoArrivalTime == b.pseudoArrivalTime)
            return a.processID < b.processID;
          return a.pseudoArrivalTime < b.pseudoArrivalTime;
        });

      auto dummy = ready.front();
      running = &dummy;
      running->state = "running";
      ready.pop_front();
    }

    if (running && running->cpuTime) {
      running->cpuTime--;
    }
    if (running)
      temp.push_back(*running);

    for (auto const &i: blocked)
      temp.push_back(i);

    for (auto const &i: ready)
      temp.push_back(i);

    if (!temp.empty())
      returnStruct.state.push_back(temp);

    cycle++;
  }
  return returnStruct;
}

#endif
```

roundRobin.hpp

```
#ifndef ROUNDROBIN
#define ROUNDROBIN

#include <algorithm>
#include <deque>
#include <iostream>
#include <list>
#include <map>
#include <vector>

#include "Process.hpp"
#include "Timeline.hpp"

Timeline roundRobin(std::vector<Process> &processes, int quanta) {
  Timeline returnStruct;
  int cycle = 0;

  std::sort(processes.begin(), processes.end(), [](auto &a, auto &b) {
    return a.arrivalTime < b.arrivalTime;
  });

  Process *running = nullptr;
  std::list<Process> blocked;
  std::deque<Process> ready;
  std::map<int, std::vector<Process> > arrivalMap;

  for (auto &i: processes) {
    arrivalMap[i.arrivalTime].push_back(i);
  }

  while (running || !blocked.empty() || !ready.empty() || arrivalMap.rbegin()-
>first > cycle) {
    std::vector<Process> temp;

    if (running && !running->cpuTime && !running->ioTime) {
      running->finishingTime = cycle - 1;
      running->turnaroundTime = running->finishingTime - running->startingTime +
1;
      running->state = "terminated";
      returnStruct.processes.push_back(*running);
      running = nullptr;
    } else if (running && !running->quanta && running->cpuTime) {
      running->state = "ready";
```



```
      running->pseudoArrivalTime = cycle;
      ready.push_back(*running);
      running = nullptr;
    } else if (running && (!running->quanta || !running->cpuTime) && running-
>ioTime) {
      running->state = "blocked";
      blocked.push_back(*running);
      running = nullptr;
    }

    if (!arrivalMap[cycle].empty()) {
      for (auto &i: arrivalMap[cycle]) {
        i.state = "ready";
        if (i.startingTime == -1)
          i.startingTime = cycle;
        ready.push_back(i);
      }
    }

    if (!blocked.empty()) {
      auto i = blocked.begin();
      while (i != blocked.end()) {
        if (i->ioTime == 0) {
          i->pseudoArrivalTime = cycle;
          i->state = "ready";
          ready.push_back(*i);
          i = blocked.erase(i);
        } else {
          i->ioTime--;
          ++i;
        }
      }
    }

    if (!running && !ready.empty()) {
      if (ready.size() > 1)
        std::sort(ready.begin(), ready.end(), [](Process &a, Process &b) {
          if (a.pseudoArrivalTime == b.pseudoArrivalTime)
            return a.processID < b.processID;
          return a.pseudoArrivalTime < b.pseudoArrivalTime;
        });

      auto dummy = ready.front();
      running = &dummy;
      running->quanta = quanta;
```



```
      running->state = "running";
      ready.pop_front();
    }

    if (running && running->cpuTime && running->quanta) {
      running->cpuTime--;
      running->quanta--;
    }

    if (running)
      temp.push_back(*running);

    for (auto const &i: blocked)
      temp.push_back(i);

    for (auto const &i: ready)
      temp.push_back(i);

    if (!temp.empty())
      returnStruct.state.push_back(temp);

    cycle++;
  }
  return returnStruct;
}

#endif
```

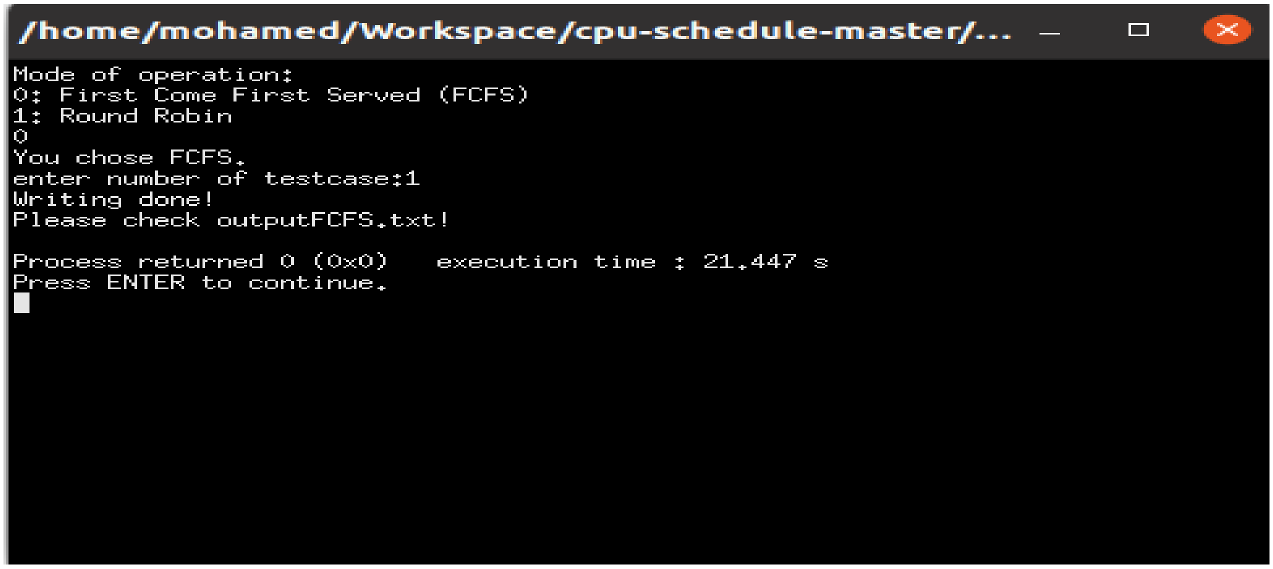
Printstats.hpp

```cpp
#ifndef PRINTSTATS
#define PRINTSTATS

#include <algorithm>
#include <iostream>

#include "Timeline.hpp"
#include "writeToFile.hpp"

std::string printStats(std::string fileName, Timeline stats) {
  std::string content;
  float activityTime = 0;

  int index = 0;

  for (auto i: stats.state) {
    std::sort(i.begin(), i.end(), [](Process a, Process b) {
      return a.processID < b.processID;
    });

    content += std::to_string(index);
    content += " ";
    for (auto const &process: i) {
      content += std::to_string(process.processID) + ": " + process.state + " ";
      if (process.state == "running")
        activityTime++;
    }
    content += "\n";
    index++;
  }

  std::string cpuUtilization = std::to_string((activityTime / stats.state.size())
* 100);
  if (*(cpuUtilization.begin() + 2) != '0')
    cpuUtilization.replace(cpuUtilization.begin() + 2, cpuUtilization.end(), "");
  else
    cpuUtilization.replace(cpuUtilization.begin() + 3, cpuUtilization.end(), "");

  content += "\nFinishing Time: " + std::to_string(stats.state.size() - 1) +
"\n";
  content += "CPU utilization: " + cpuUtilization + "%\n";
  std::sort(stats.processes.begin(), stats.processes.end(), [](Process a, Process b) {
    return a.processID < b.processID;
  });

  for (auto const &i: stats.processes) {
    content += "Turnaround time of process " + std::to_string(i.processID) + ": ";
    content += std::to_string(i.turnaroundTime) + "\n";
  }

  writeToFile(fileName, content);
  std::cout << "Writing done!\n"
            << "Please check " << fileName << "!"
            << "\n";

  return content;
}

#endif
```

writeToFile.hpp

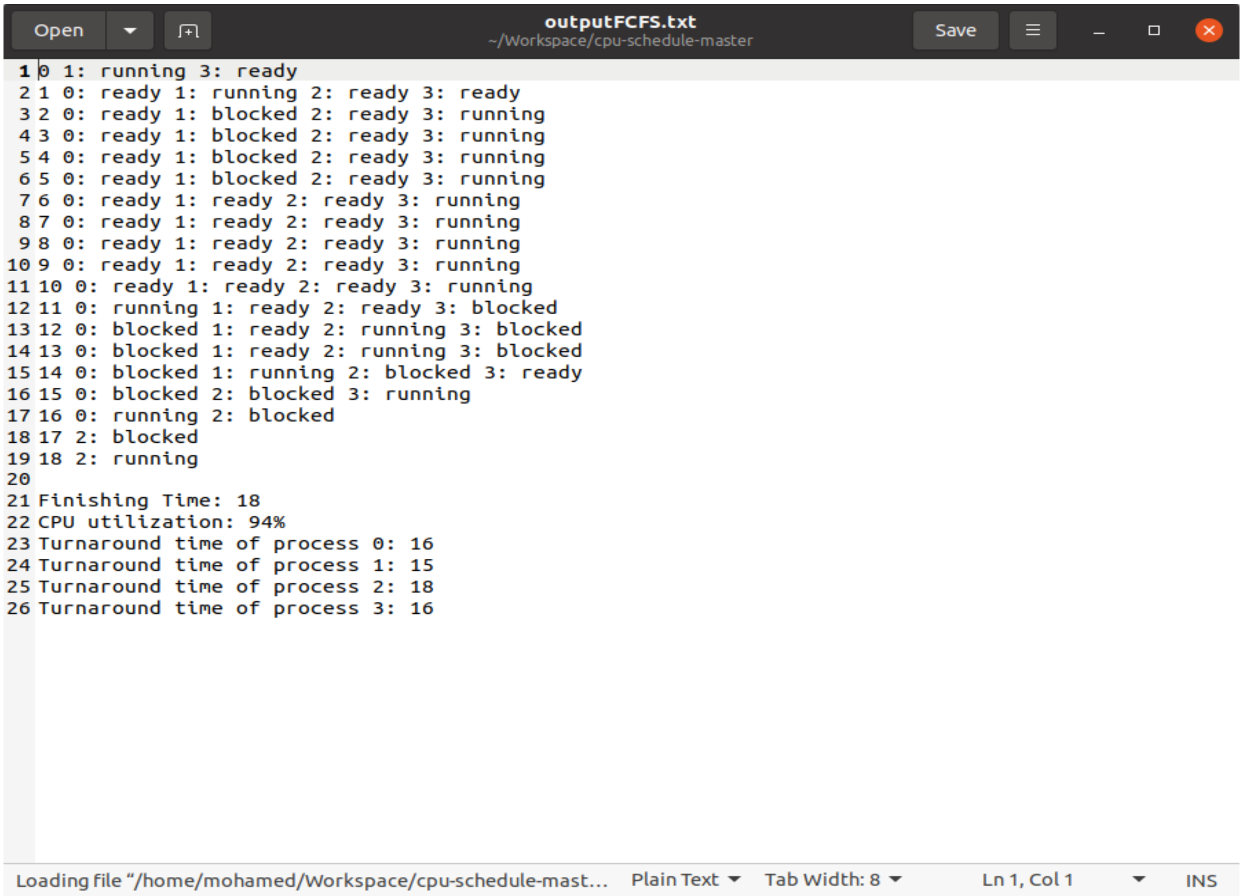
```
#ifndef WRITETOFILE
#define WRITETOFILE

#include <fstream>
#include <iostream>

#include "Process.hpp"

bool writeToFile(std::string filename, std::string content) {
  remove(filename.c_str());
  std::ofstream outputFile;
  outputFile.open(filename);
  outputFile << content;
  outputFile.close();
  return true;
}

#endif
```

FCFS:



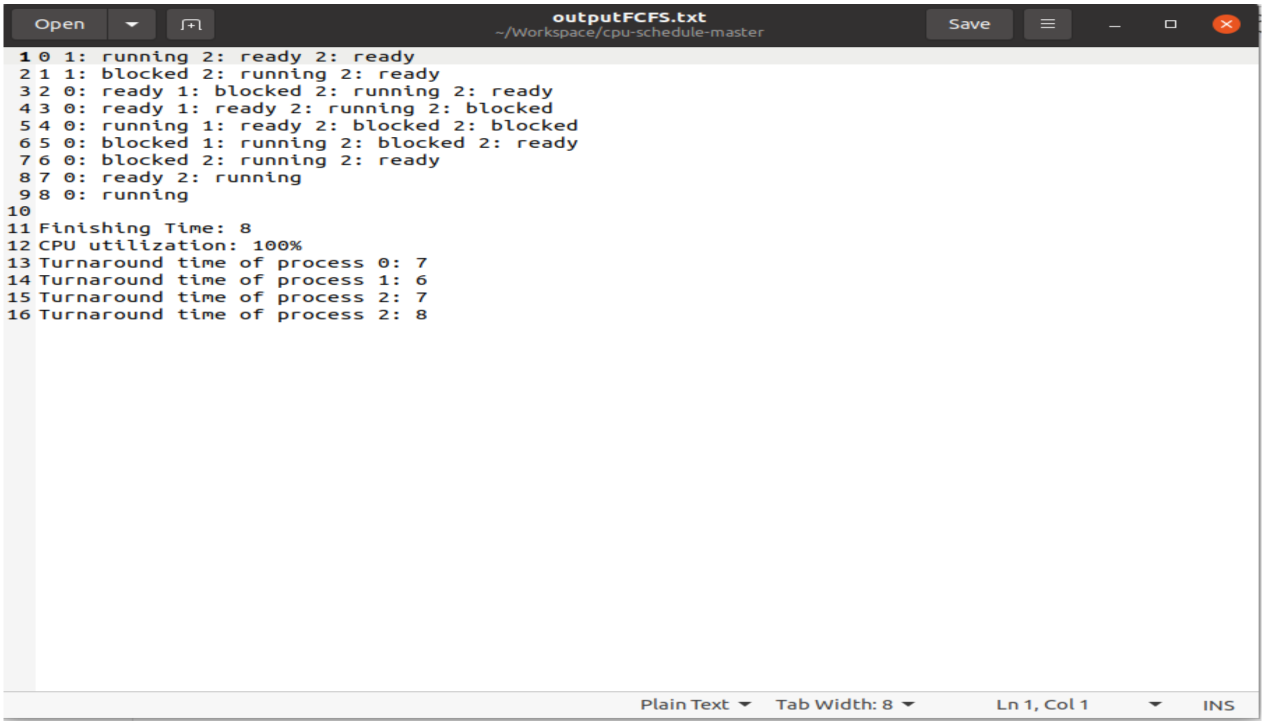
```
Mode of operation:
0: First Come First Served (FCFS)
1: Round Robin
0
You chose FCFS.
enter number of testcase:1
Writing done!
Please check outputFCFS.txt!

Process returned 0 (0x0)   execution time : 21.447 s
Press ENTER to continue.
```

Output of testcase 1:


Open
outputFCFS.txt
~/Workspace/cpu-schedule-master
Save
Loading file "/home/mohamed/Workspace/cpu-schedule-mast...
Plain Text
Tab Width: 8
Ln 1, Col 1
INS

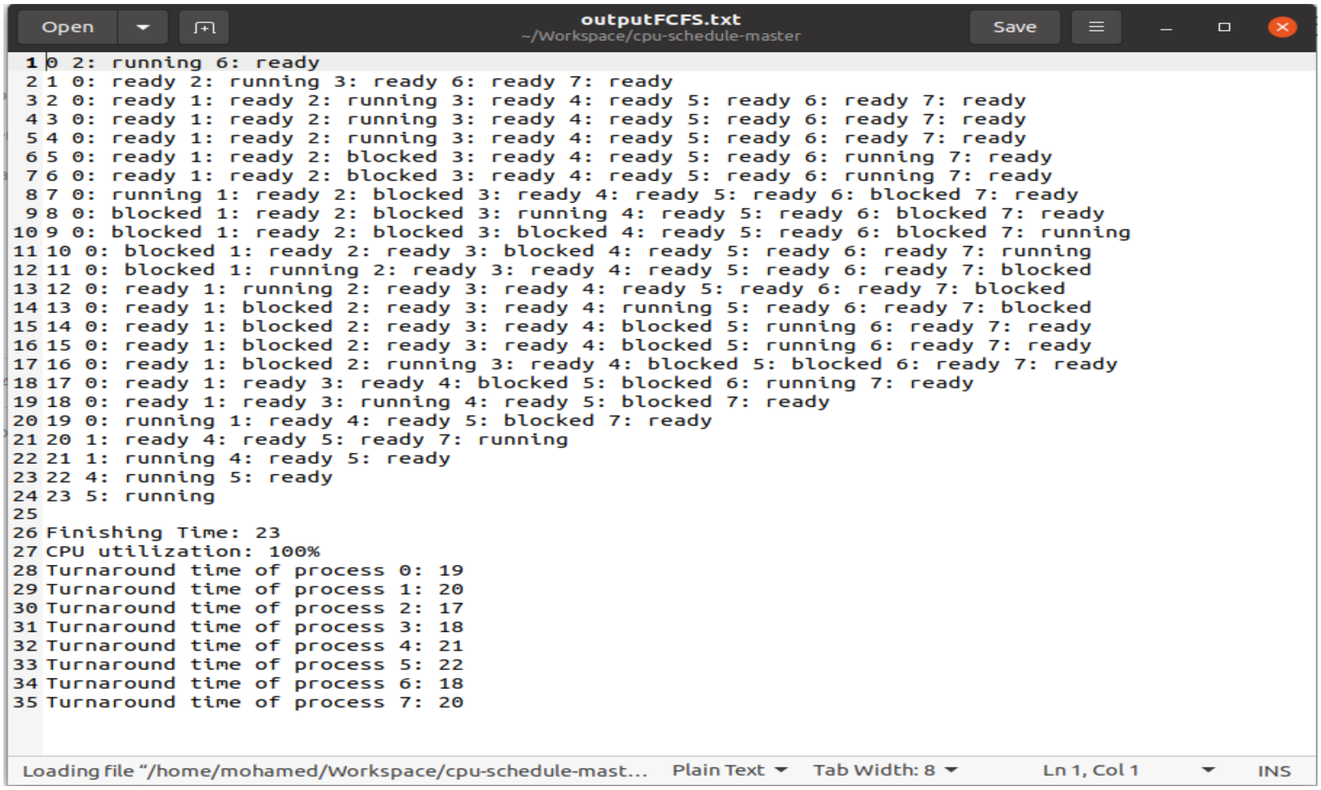
```
0 1: running 3: ready
1 0: ready 1: running 2: ready 3: ready
2 0: ready 1: blocked 2: ready 3: running
3 0: ready 1: blocked 2: ready 3: running
4 0: ready 1: blocked 2: ready 3: running
5 0: ready 1: blocked 2: ready 3: running
6 0: ready 1: ready 2: ready 3: running
7 0: ready 1: ready 2: ready 3: running
8 0: ready 1: ready 2: ready 3: running
9 0: ready 1: ready 2: ready 3: running
10 0: ready 1: ready 2: ready 3: running
11 0: running 1: ready 2: ready 3: blocked
12 0: blocked 1: ready 2: running 3: blocked
13 0: blocked 1: ready 2: running 3: blocked
14 0: blocked 1: running 2: blocked 3: ready
15 0: blocked 2: blocked 3: running
16 0: running 2: blocked
17 2: blocked
18 2: running

Finishing Time: 18
CPU utilization: 94%
Turnaround time of process 0: 16
Turnaround time of process 1: 15
Turnaround time of process 2: 18
Turnaround time of process 3: 16
```

output of testcase 2:



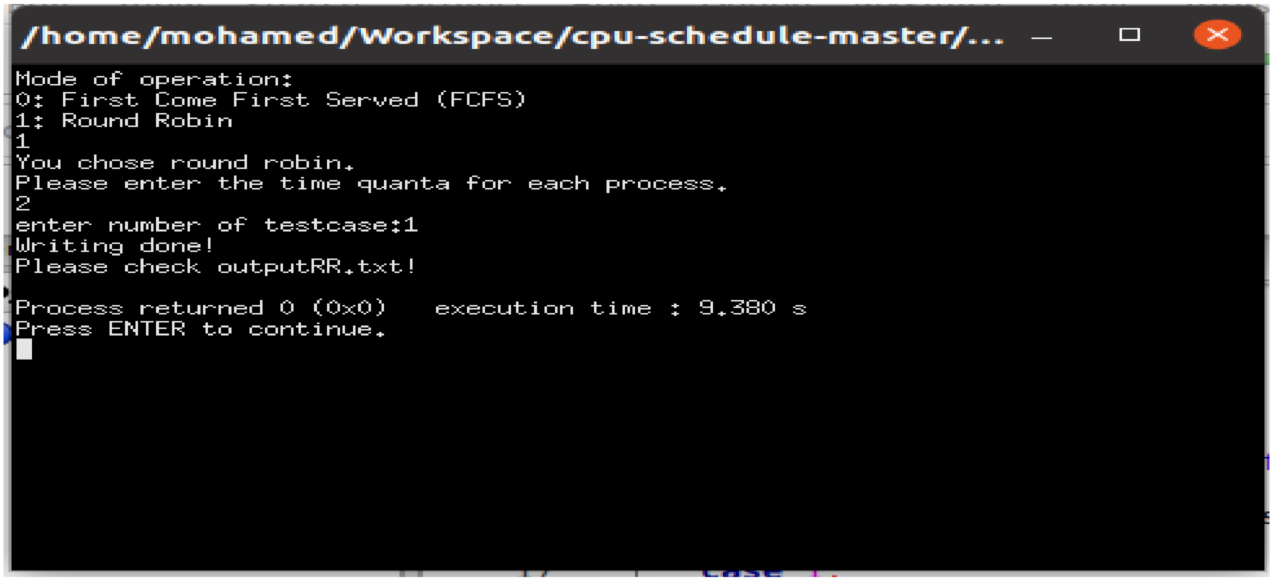
```
0 1: running 2: ready 2: ready
1 1: blocked 2: running 2: ready
2 0: ready 1: blocked 2: running 2: ready
3 0: ready 1: ready 2: running 2: blocked
4 0: running 1: ready 2: blocked 2: blocked
5 0: blocked 1: running 2: blocked 2: ready
6 0: blocked 2: running 2: ready
7 0: ready 2: running
8 0: running

Finishing Time: 8
CPU utilization: 100%
Turnaround time of process 0: 7
Turnaround time of process 1: 6
Turnaround time of process 2: 7
Turnaround time of process 2: 8
```

Output of testcase 3:


Open
outputFCFS.txt
~/Workspace/cpu-schedule-master
Save
Loading file "/home/mohamed/Workspace/cpu-schedule-mast...
Plain Text
Tab Width: 8
Ln 1, Col 1
INS

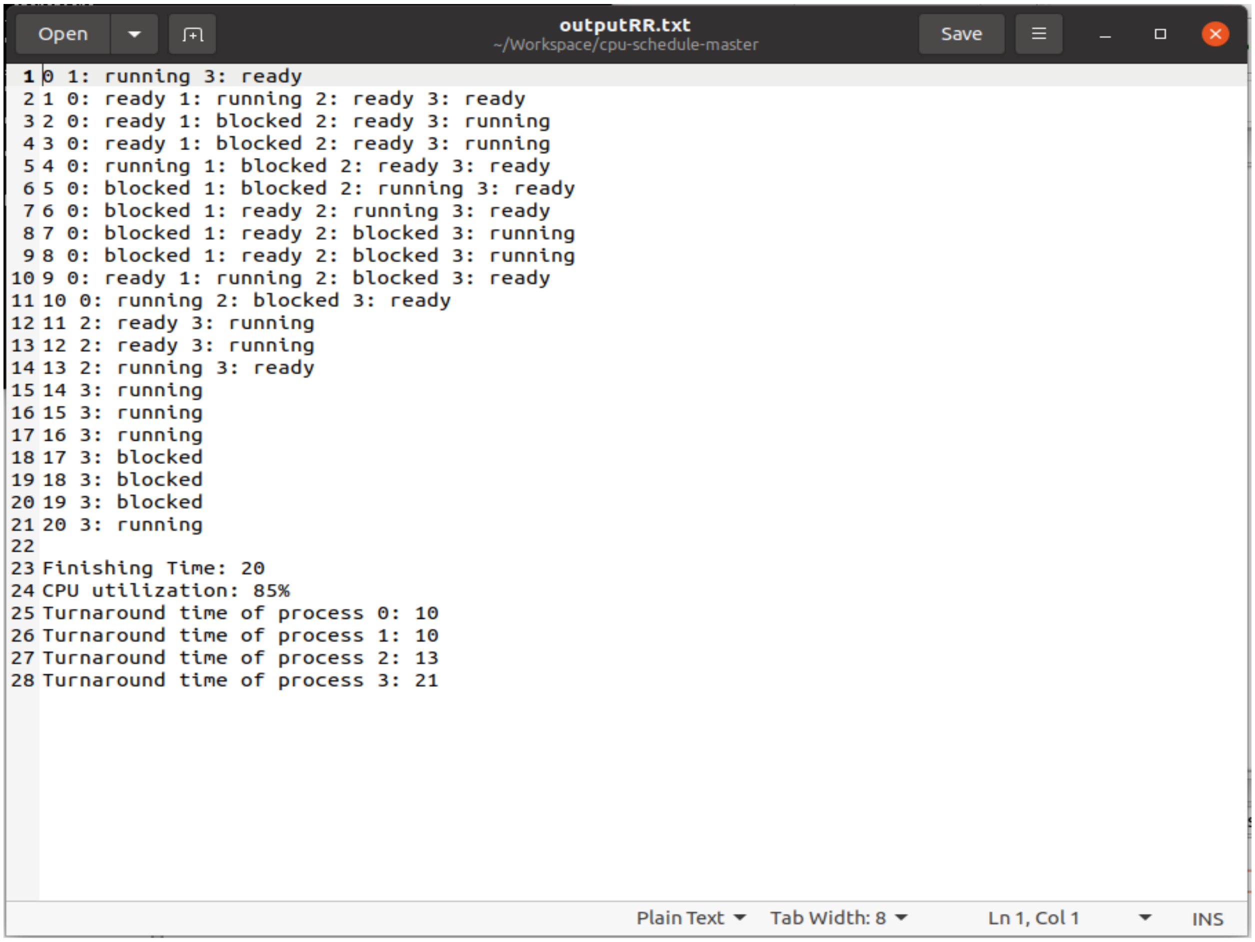
```
0 2: running 6: ready
1 0: ready 2: running 3: ready 6: ready 7: ready
2 0: ready 1: ready 2: running 3: ready 4: ready 5: ready 6: ready 7: ready
3 0: ready 1: ready 2: running 3: ready 4: ready 5: ready 6: ready 7: ready
4 0: ready 1: ready 2: running 3: ready 4: ready 5: ready 6: ready 7: ready
5 0: ready 1: ready 2: blocked 3: ready 4: ready 5: ready 6: running 7: ready
6 0: ready 1: ready 2: blocked 3: ready 4: ready 5: ready 6: running 7: ready
7 0: running 1: ready 2: blocked 3: ready 4: ready 5: ready 6: blocked 7: ready
8 0: blocked 1: ready 2: blocked 3: running 4: ready 5: ready 6: blocked 7: ready
9 0: blocked 1: ready 2: blocked 3: blocked 4: ready 5: ready 6: blocked 7: running
10 0: blocked 1: ready 2: ready 3: blocked 4: ready 5: ready 6: ready 7: running
11 0: blocked 1: running 2: ready 3: ready 4: ready 5: ready 6: ready 7: blocked
12 0: ready 1: running 2: ready 3: ready 4: ready 5: ready 6: ready 7: blocked
13 0: ready 1: blocked 2: ready 3: ready 4: running 5: ready 6: ready 7: blocked
14 0: ready 1: blocked 2: ready 3: ready 4: blocked 5: running 6: ready 7: ready
15 0: ready 1: blocked 2: ready 3: ready 4: blocked 5: running 6: ready 7: ready
16 0: ready 1: blocked 2: running 3: ready 4: blocked 5: blocked 6: ready 7: ready
17 0: ready 1: ready 3: ready 4: blocked 5: blocked 6: running 7: ready
18 0: ready 1: ready 3: running 4: ready 5: blocked 7: ready
19 0: running 1: ready 4: ready 5: blocked 7: ready
20 1: ready 4: ready 5: ready 7: running
21 1: running 4: ready 5: ready
22 4: running 5: ready
23 5: running

Finishing Time: 23
CPU utilization: 100%
Turnaround time of process 0: 19
Turnaround time of process 1: 20
Turnaround time of process 2: 17
Turnaround time of process 3: 18
Turnaround time of process 4: 21
Turnaround time of process 5: 22
Turnaround time of process 6: 18
Turnaround time of process 7: 20
```

RoundRobin(q=2):



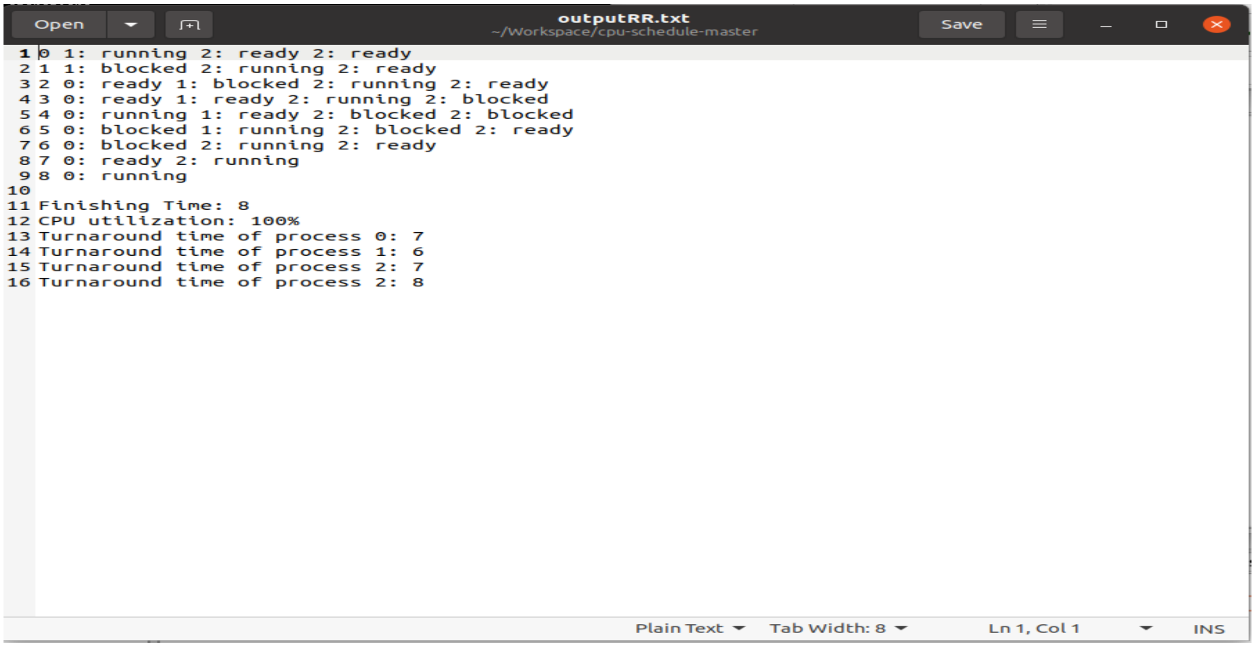
```
Mode of operation:
0: First Come First Served (FCFS)
1: Round Robin
1
You chose round robin.
Please enter the time quanta for each process.
2
enter number of testcase:1
Writing done!
Please check outputRR.txt!

Process returned 0 (0x0)   execution time : 9.380 s
Press ENTER to continue.
```

output of testcase 1:



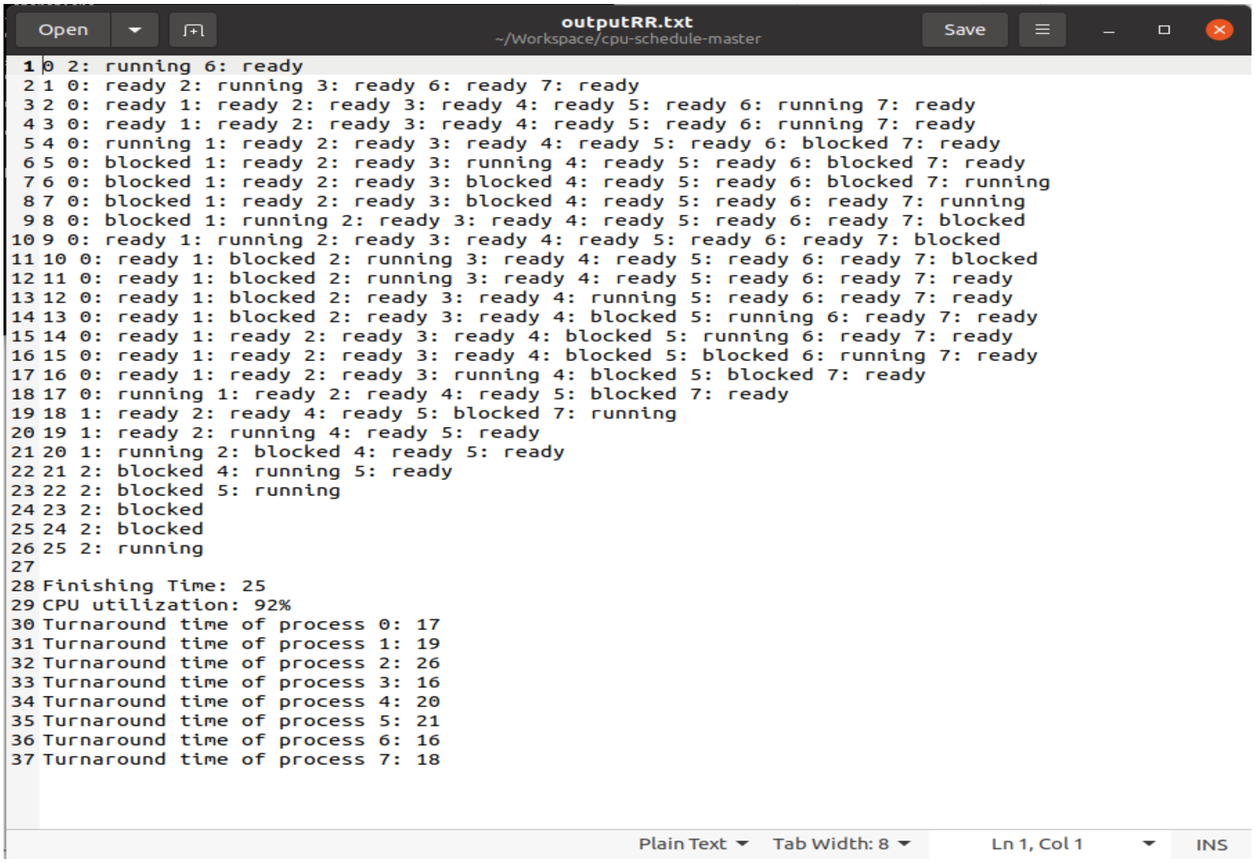
```
0 1: running 3: ready
1 0: ready 1: running 2: ready 3: ready
2 0: ready 1: blocked 2: ready 3: running
3 0: ready 1: blocked 2: ready 3: running
4 0: running 1: blocked 2: ready 3: ready
5 0: blocked 1: blocked 2: running 3: ready
6 0: blocked 1: ready 2: running 3: ready
7 0: blocked 1: ready 2: blocked 3: running
8 0: blocked 1: ready 2: blocked 3: running
9 0: ready 1: running 2: blocked 3: ready
10 0: running 2: blocked 3: ready
11 2: ready 3: running
12 2: ready 3: running
13 2: running 3: ready
14 3: running
15 3: running
16 3: running
17 3: blocked
18 3: blocked
19 3: blocked
20 3: running

Finishing Time: 20
CPU utilization: 85%
Turnaround time of process 0: 10
Turnaround time of process 1: 10
Turnaround time of process 2: 13
Turnaround time of process 3: 21
```

output of testcase 2:



```
0 1: running 2: ready 2: ready
1 1: blocked 2: running 2: ready
2 0: ready 1: blocked 2: running 2: ready
3 0: ready 1: ready 2: running 2: blocked
4 0: running 1: ready 2: blocked 2: blocked
5 0: blocked 1: running 2: blocked 2: ready
6 0: blocked 2: running 2: ready
7 0: ready 2: running
8 0: running

Finishing Time: 8
CPU utilization: 100%
Turnaround time of process 0: 7
Turnaround time of process 1: 6
Turnaround time of process 2: 7
Turnaround time of process 2: 8
```

output of testcase 3:



```
0 2: running 6: ready
1 0: ready 2: running 3: ready 6: ready 7: ready
2 0: ready 1: ready 2: ready 3: ready 4: ready 5: ready 6: running 7: ready
3 0: ready 1: ready 2: ready 3: ready 4: ready 5: ready 6: running 7: ready
4 0: running 1: ready 2: ready 3: ready 4: ready 5: ready 6: blocked 7: ready
5 0: blocked 1: ready 2: ready 3: running 4: ready 5: ready 6: blocked 7: ready
6 0: blocked 1: ready 2: ready 3: blocked 4: ready 5: ready 6: blocked 7: running
7 0: blocked 1: ready 2: ready 3: blocked 4: ready 5: ready 6: ready 7: running
8 0: blocked 1: running 2: ready 3: ready 4: ready 5: ready 6: ready 7: blocked
9 0: ready 1: running 2: ready 3: ready 4: ready 5: ready 6: ready 7: blocked
10 0: ready 1: blocked 2: running 3: ready 4: ready 5: ready 6: ready 7: blocked
11 0: ready 1: blocked 2: running 3: ready 4: ready 5: ready 6: ready 7: ready
12 0: ready 1: blocked 2: ready 3: ready 4: running 5: ready 6: ready 7: ready
13 0: ready 1: blocked 2: ready 3: ready 4: blocked 5: running 6: ready 7: ready
14 0: ready 1: ready 2: ready 3: ready 4: blocked 5: running 6: ready 7: ready
15 0: ready 1: ready 2: ready 3: ready 4: blocked 5: blocked 6: running 7: ready
16 0: ready 1: ready 2: ready 3: running 4: blocked 5: blocked 7: ready
17 0: running 1: ready 2: ready 4: ready 5: blocked 7: ready
18 1: ready 2: ready 4: ready 5: blocked 7: running
19 1: ready 2: running 4: ready 5: ready
20 1: running 2: blocked 4: ready 5: ready
21 2: blocked 4: running 5: ready
22 2: blocked 5: running
23 2: blocked
24 2: blocked
25 2: running

Finishing Time: 25
CPU utilization: 92%
Turnaround time of process 0: 17
Turnaround time of process 1: 19
Turnaround time of process 2: 26
Turnaround time of process 3: 16
Turnaround time of process 4: 20
Turnaround time of process 5: 21
Turnaround time of process 6: 16
Turnaround time of process 7: 18
```