```
/***************************************************************************
/
/      filename:  Output.cpp
/
/   description:  Implements Event D for the simulator
/
/        author:  Paladino, Zac
/      login id:  cps346−n1.16
/
/         class:  CPS 346
/    instructor:  Perugini
/    assignment:  PJ #2
/
/      assigned:  February 18, 2009
/           due:  March 11, 2009
/
/**************************************************************************/

#include <iostream>
#include <iomanip>
#include <queue>
#include <list>
#include <vector>
#include <string>
#include <fstream>
using namespace std;
#include "Functions.h"

struct Process
{
  string Event;
  string RQ;
  int Time, Job, Memory, RT, RTM, RQT, FTime, STime, IOBurst, IOS, IOB;
  bool started, IOClean;
};

struct Semephore
{
  int value;
    list < Process > SemList;
};

void
EventD (vector < string > tokens, list < Process > JobQ, list < Process > RQ1,
        list < Process > RQ2, list < Process > CPU,
        vector < Process > Finished, vector < Process > IO, int memory,
        Semephore Semephores[], bool & getcm, int time, ofstream & out,
        int CPURQ1, int CPURQ2)
{
  if (tokens[0] == "D") {
    if (time == StringToInt (tokens[1])) {

      out << "Event: D  " << "Time: " << time << endl;
      out << endl;
      out << "************************************************************" <<
        endl;
      out << endl;
      out << "The status of the simulator at time " << time << "." << endl;
      out << endl;
      out << "The contents of the JOB SCHEDULING QUEUE" << endl;
      out << "−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−" << endl;
      out << endl;
      out << "Job #  Arr. Time  Mem. Req.  Run Time" << endl;
      out << "−−−−− −−−−−−−−− −−−−−−−−− −−−−−−−−" << endl;
      out << endl;
      if (!JobQ.empty ()) {
        list < Process >::iterator i = JobQ.begin (), j = JobQ.end ();
        for (; i != j; i++) {
          out << setw (5) << (*i).Job << " " << setw (9) << (*i).
```

```
            Time << " " << setw (9) << (*i).
            Memory << " " << setw (8) << (*i).RT << endl;
        }
      }
      else {
        out << "The Job Scheduling Queue is empty." << endl;
      }
      out << endl;
      out << "The contents of the FIRST LEVEL READY QUEUE" << endl;
      out << "−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−" << endl;
      out << endl;
      if (!RQ1.empty ()) {
        list < Process >::iterator i = RQ1.begin (), j = RQ1.end ();
        for (; i != j; i++) {
          out << setw (5) << (*i).Job << " " << setw (9) << (*i).
            Time << " " << setw (9) << (*i).
            Memory << " " << setw (8) << (*i).RT << endl;
        }
      }
      else {
        out << "The First Level Ready Queue is empty." << endl;
      }
      out << endl;
      out << endl;
      out << "The contents of the SECOND LEVEL READY QUEUE" << endl;
      out << "−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−" << endl;
      out << endl;
      if (!RQ2.empty ()) {
        list < Process >::iterator i = RQ2.begin (), j = RQ2.end ();
        for (; i != j; i++) {
          out << setw (5) << (*i).Job << " " << setw (9) << (*i).
            Time << " " << setw (9) << (*i).
            Memory << " " << setw (8) << (*i).RT << endl;
        }
      }
      else {
        out << "The Second Level Ready Queue is empty." << endl;
      }
      out << endl;
      out << endl;
      out << "The contents of the I/O WAIT QUEUE" << endl;
      out << "−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−" << endl;
      out << endl;
      if (!IO.empty ()) {
        out <<
          "Job #  Arr. Time  Mem. Req.  Run Time  IO Start Time IO Burst  Comp. Time"
          << endl;
        out <<
          "−−−−− −−−−−−−−− −−−−−−−−− −−−−−−−− −−−−−−−−−−−−− −−−−−−−− −−−−−−−−−−"
          << endl;
        out << endl;
        for (int i = 0; i < (static_cast < int >(IO.size ())); i++) {
          out << setw (5) << IO[i].Job << " " << setw (9) << IO[i].
            Time << " " << setw (9) << IO[i].
            Memory << " " << setw (8) << IO[i].
            RT << " " << setw (13) << IO[i].
            IOS << " " << setw (9) << IO[i].IOB << " " << setw (10) << (IO[i].
                                                                        IOB +
                                                                        IO[i].
                                                                        IOS)
            << endl;
      }}
      else {
        out << "The I/O Waiting Queue is empty." << endl;
      }
      out << endl;
      out << endl;
      out << "The contents of the SEMAPHORE ZERO" << endl;
      out << "−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−" << endl;
      out << endl;
      out << "The value of semaphore 0 is " << Semephores[0].
        value << "." << endl;
      out << endl;
      if (!Semephores[0].SemList.empty ()) {
        list < Process >::iterator i = Semephores[0].SemList.begin (), j =
          Semephores[0].SemList.end ();
        for (; i != j; i++) {
          out << (*i).Job << endl;
        }
      }
      else {
        out << "The waiting list for semaphore 0 is empty." << endl;
      }
      out << endl;
      out << endl;
      out << "The contents of the SEMAPHORE ONE" << endl;
      out << "--------------------------------" << endl;
      out << endl;
      out << "The value of semaphore 1 is " << Semephores[1].
        value << "." << endl;
      out << endl;
      if (!Semephores[1].SemList.empty ()) {
        list < Process >::iterator i = Semephores[1].SemList.begin (), j =
          Semephores[1].SemList.end ();
        for (; i != j; i++) {
          out << (*i).Job << endl;
        }
      }
      else {
        out << "The waiting list for semaphore 1 is empty." << endl;
      }
      out << endl;
      out << endl;
      out << "The contents of the SEMAPHORE TWO" << endl;
      out << "--------------------------------" << endl;
      out << endl;
      out << "The value of semaphore 2 is " << Semephores[2].
        value << "." << endl;
      out << endl;
      if (!Semephores[2].SemList.empty ()) {
        list < Process >::iterator i = Semephores[2].SemList.begin (), j =
          Semephores[2].SemList.end ();
        for (; i != j; i++) {
          out << (*i).Job << endl;
        }
      }
      else {
        out << "The waiting list for semaphore 2 is empty." << endl;
      }
      out << endl;
      out << endl;
      out << "The contents of the SEMAPHORE THREE" << endl;
      out << "----------------------------------" << endl;
      out << endl;
      out << "The value of semaphore 3 is " << Semephores[3].
        value << "." << endl;
      out << endl;
      if (!Semephores[3].SemList.empty ()) {
        list < Process >::iterator i = Semephores[3].SemList.begin (), j =
          Semephores[3].SemList.end ();
        for (; i != j; i++) {
          out << (*i).Job << endl;
        }
      }
      else {
        out << "The waiting list for semaphore 3 is empty." << endl;
      }
      out << endl;
```

```
      out << endl;
      out << "The contents of the SEMAPHORE FOUR" << endl;
      out << "--------------------------------" << endl;
      out << endl;
      out << "The value of semaphore 4 is " << Semephores[4].
        value << "." << endl;
      out << endl;
      if (!Semephores[4].SemList.empty ()) {
        list < Process >::iterator i = Semephores[4].SemList.begin (), j =
          Semephores[4].SemList.end ();
        for (; i != j; i++) {
          out << (*i).Job << endl;
        }
      }
      else {
        out << "The waiting list for semaphore 4 is empty." << endl;
      }
      out << endl;
      out << endl;
      out << "The CPU    Start Time    CPU burst time left" << endl;
      out << "-------  ----------  ------------------" << endl;
      out << endl;
      if (!CPU.empty ()) {
        out << setw (7) << CPU.front ().Job << setw (10) << "  " << CPU.
          front ().STime << setw (19) << "  " << CPU.front ().RTM << endl;
        //if (CPU.front ().RQ == "RQ1") {
        //out << setw (19) << CPURQ1 << endl;
        //}
        //else {
        //out << setw (19) << CPURQ2 << endl;
        //}
      }
      else {
        out << "The CPU is idle." << endl;
      }
      out << endl;
      out << endl;
      out << "The contents of the FINISHED LIST" << endl;
      out << "-------------------------------" << endl;
      out << endl;
      out << "Job #  Arr. Time  Mem. Req.  Run Time  Start Time   Com. Time"
        << endl;
      out << "----- --------- --------- -------- ----------  ---------"
        << endl;
      out << endl;
      for (int i = 0; i < (static_cast < int >(Finished.size ())); i++) {
        out << setw (5) << Finished[i].Job << " " << setw (9) << Finished[i].
          Time << " " << setw (9) << Finished[i].
          Memory << " " << setw (8) << Finished[i].
          RT << " " << setw (10) << Finished[i].
          STime << "  " << setw (9) << Finished[i].FTime << endl;
      }
      out << endl;
      out << endl;
      out << "There are " << memory <<
        " blocks of main memory available in the system." << endl;
      out << endl;
      getcm = true;
    }
    else {
      getcm = false;
    }
  }

}
```