```
/*****************************************************************************
/
/      filename:  CPU.cpp
/
/   description:  Implements the CPU 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;
};

bool
DealCPU (list < Process > &CPU, list < Process > &RQ1,
         list < Process > &RQ2, vector < Process > &Finished, int &CPURQ1,
         int &CPURQ2, int &memory, int &time, ofstream & out)
{
  bool cpun = false;
  if (CPU.empty ()) {
    if (!RQ1.empty ()) {
      Process temp = RQ1.front ();
      CPU.push_back (temp);
      cpun = true;
      RQ1.pop_front ();
    }
    else if (!RQ2.empty ()) {
      Process temp = RQ2.front ();
      CPU.push_back (temp);
      cpun = true;
      RQ2.pop_front ();
    }
  }
  if (!CPU.empty ()) {
    if (cpun == true && !CPU.front ().started) {
      CPU.front ().STime = time;
      CPU.front ().started = true;
      cpun = false;
    }
    if (CPU.front ().RQ == "RQ1") {
```



```
      if (CPU.front ().RTM > 0) {
        if (CPURQ1 > 0) {
          CPU.front ().RTM--;
          CPURQ1--;
        }
        else {
          Process temp = CPU.front ();
          temp.RQ = "RQ2";
          RQ2.push_back (temp);
          CPU.pop_front ();
          CPURQ1 = 100;
          cpun = true;
          out << "Event: E  " << "Time: " << time << endl;
          return true;
        }
      }
      else {
        Process temp = CPU.front ();
        temp.FTime = time;
        memory += temp.Memory;
        Finished.push_back (temp);
        CPU.pop_front ();
        CPURQ1 = 100;
        cpun = true;
        out << "Event: T  " << "Time: " << time << endl;
        return true;
      }
    }
    else {
      if (!RQ1.empty ()) {
        CPURQ2 = 300;
        CPURQ1 = 100;
        RQ2.push_back (CPU.front ());
        CPU.pop_front ();
        CPU.push_back (RQ1.front ());
        RQ1.pop_front ();
        CPU.front ().RTM--;
        CPURQ1--;
      }
      else {
        if (CPU.front ().RTM > 0) {
          if (CPURQ2 > 0) {
            CPU.front ().RTM--;
            CPURQ2--;
          }
          else {
            Process temp = CPU.front ();
            RQ2.push_back (temp);
            CPU.pop_front ();
            CPURQ2 = 300;
            cpun = true;
            out << "Event: E  " << "Time: " << time << endl;
            return true;
          }
        }
        else {
          Process temp = CPU.front ();
          temp.FTime = time;
          memory += temp.Memory;
          Finished.push_back (temp);
          CPU.pop_front ();
          CPURQ2 = 300;
          cpun = true;
          out << "Event: T  " << "Time: " << time << endl;
          return true;
        }
      }
    }
  }
```

```
  return false;
}
```