```
/*****************************************************************************
/
/      filename:  IO.cpp
/
/   description:  Implements the I/O 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
HandleIO (vector < Process > &IO, list < Process > &CPU,
          list < Process > &RQ1, vector < string > &tokens, int &time,
          bool & getcm, list < Process > &RQ2, vector < Process > &Finished,
          int &CPURQ1, int &CPURQ2, int &memory, ofstream & out, bool & NOGO)
{
  if (!IO.empty ()) {
    for (int i = 0; i < IO.size (); i++) {
      if (IO[i].IOBurst > 0) {
        IO[i].IOBurst−−;
      }
      if (IO[i].IOBurst == 0) {
        IO[i].RQ = "RQ1";
        IO[i].IOClean = true;
        RQ1.push_back (IO[i]);
        out << "Event: C  " << "Time: " << time << endl;
      }
    }
    int j = static_cast < int >(IO.size ());
    vector < Process > temp;
    for (int i = 0; i < j; i++) {
      if (!IO[i].IOClean) {
        temp.push_back (IO[i]);
      }
    }
    IO.clear ();
    j = static_cast < int >(temp.size ());
```

```
    for (int i = 0; i < j; i++) {
      IO.push_back (temp[i]);
    }
  }

  if (tokens[0] == "I") {
    if (time == StringToInt (tokens[1])) {
      if (!CPU.empty ()) {
        CPU.front ().IOBurst = StringToInt (tokens[2]);
        CPU.front ().IOClean = false;
        CPU.front ().IOS = time;
        CPU.front ().IOB = StringToInt (tokens[2]);
        IO.push_back (CPU.front ());
        out << "Event: I  " << "Time: " << time << endl;
        CPU.pop_front ();
        CPURQ1 = 100;
        CPURQ2 = 300;
      }
      getcm = true;
    }
    else {
      getcm = false;
    }

  }

}
```