```
/*****************************************************************************
//
//      filename:  Job.cpp
//
//   description:  Implements the Job Scheduler 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
HandleJob (list < Process > &JobQ, list < Process > &RQ1,
           vector < string > &tokens, int &time, bool & ev, bool & eve,
           bool & getcm, int &memory, int &count, int &tot_proc)
{
  if (tokens[0] == "A") {
    if (count == 0) {
      time = StringToInt (tokens[1]);
      count++;
    }
    if (time == StringToInt (tokens[1])) {
      Process newp;
      newp.Event = "A";
      newp.RQ = "RQ1";
      newp.Time = StringToInt (tokens[1]);
      newp.Job = StringToInt (tokens[2]);
      newp.Memory = StringToInt (tokens[3]);
      newp.RT = StringToInt (tokens[4]);
      newp.RTM = StringToInt (tokens[4]);
      newp.started = false;
      newp.IOBurst = 0;
      newp.IOClean = false;
      newp.RQT = 0;
      ev = true;
      if (newp.Memory <= 512) {
        JobQ.push_back (newp);
        tot_proc++;
```



```
      }
      else {
        eve = true;
      }
      getcm = true;
    }
    else {
      getcm = false;
    }
  }
  while (!JobQ.empty ()) {
    if (JobQ.front ().Memory <= memory) {
      JobQ.front ().RQT = time;
      RQ1.push_back (JobQ.front ());
      memory -= JobQ.front ().Memory;
      JobQ.pop_front ();
    }
    else {
      break;
    }
  }


}
```