```
/*--- hiscore.cpp ---*/
//Mike Hanling
//hiscore.cpp


#include <iostream>
#include <string>
#include <vector>
using namespace std;


struct result {
  string name;
  int score;
};


istream& operator>> (istream& in, result& res);
bool operator< (result a, result b);
ostream& operator<< (ostream& out, result res);


int main() {
  int n = 5;
  cout << "Enter " << n << " results:" << endl;
  vector<result> res;

  for (int i=0; i < n; ++i) {
    result temp;
    cin >> temp;
    res.push_back(temp);
  }

  // find highest score
  result best = res[0];
  for (int i=1; i < res.size(); ++i) {
    if (best < res[i]) {
      best = res[i];
    }
  }

  // print highest score
  cout << "The best result is " << best << endl;

  return 0;
}


istream& operator>> (istream& in, result& res) {
  in >> res.name >> res.score;
  return in;
}

bool operator< (result a, result b) {
  return a.score < b.score;
}

ostream& operator<< (ostream& out, result res) {
  out << res.name << " (" << res.score << " points)" << endl;
  return out;
}

/*--- sched.cpp ---*/
//Mike Hanling
//sched.cpp


#include <iostream>
#include <string>
#include <vector>
#include <fstream>
using namespace std;


struct section{
  string course;
  int section;
  string meets;
};


istream& operator>> (istream& in, section& sec);
ostream& operator<< (ostream& out, section sec);
bool overlaps(string pat, char day, int per);

int main () {
  //open file
  cout << "file: ";
  string filename;
  cin >> filename;
  ifstream fin(filename.c_str());

  //read in data
  int size = 0;
  char junk;
  fin >> junk >> junk >> size;
  vector<section> classes;
  for (int i = 0; i < size; i++) {
    section temp;
    fin >> temp;
    classes.push_back(temp);
  }

  //takes commands
  string cmd;
  while (1) {
    cout << "command: ";
    cin >> cmd;
    if (cmd == "quit") break;

    char day;
    int per;
    cin >> day >> per;

    //print out overlaps
    for (int i = 0; i < size; i++) {
      if (overlaps(classes[i].meets, day, per)) {
        cout << classes[i];
      }
    }
  }
  return 0;
}


istream& operator>> (istream& in, section& sec) {
  in >> sec.course >> sec.section >> sec.meets;
  return in;
}

ostream& operator<< (ostream& out, section sec) {
  out << sec.course << ' ' << sec.section << ' '<< sec.meets << endl;
  return out;
}


/*****************************************************************************
```

```
 Input: pat - a string representing a meeting time,
              e.g. "MWF2,R34" or "TR9" or "MF5,T65"
        day - a char, one of M,T,W,R,F
        per - an int, one of the regular periods, i.e. 1,2,3,4,5,6
 Output:true if the meeting time in pattern pat overlaps with period day,per,
        false otherwise
        Ex1 - overlaps("MWF2,R12",'M',8) -> true
        Ex2 - overlaps("TR10",'T',4) -> false
*****************************************************************************/
bool overlaps(string pat, char day, int per) {
  bool dflag = false; // day match flag
  for(int i = 0; i < pat.length(); ++i) {
    if (pat[i] == ',') {
      dflag = false;
    } else if ('A' <= pat[i] && pat[i] <= 'Z') {
      dflag = dflag || pat[i] == day;
    } else {
      int q;
      if (pat[i] == '1' && i+1 < pat.length() && pat[i+1] == '0') {
        q = 10;
      } else {
        q = (pat[i] - '0');
      }
      if (dflag && (per == q || (per-1)/2 + 8 == q)) {
        return true;
      }
    }
  }
  return false;
}
```