```
#include<iostream>
#include<cassert>
#include<fstream>

using namespace std;

// This program computes your course grade!


const int NUM_EXAM=3;
const int NUM_HW=5;
const int NUM_INLAB=13;
const int NUM_PROJECT=6;
const int NUM_PRACTICAL=2;
const int NUM_QUIZ=2;

const int WEIGHT_EXAM     = 50;
const int WEIGHT_HW       = 5;
const int WEIGHT_INLAB    = 6;
const int WEIGHT_PROJECT  = 27;
const int WEIGHT_PRACTICAL= 8;
const int WEIGHT_QUIZ     = 4;

const int OUTOF_EXAM     = 100;
const int OUTOF_HW       = 50;
const int OUTOF_INLAB    = 30;
const int OUTOF_PROJECT  = 100;
const int OUTOF_PRACTICAL= 100;
const int OUTOF_QUIZ     = 30;


struct Grades {
    int exam[NUM_EXAM];
    int hw[NUM_HW];
    int inlab[NUM_INLAB];
    int project[NUM_PROJECT];
    int practical[NUM_PRACTICAL];
    int quiz[NUM_QUIZ];
};


void sort(int list[], int num)
{
    int i=0;
    int j=0;
    int tmp;
    while(i<num-1)
    {
        while(j<num-1)
        {
            if(list[j]>list[j+1])
            {
                tmp=list[j];
                list[j]=list[j+1];
                list[j+1]=tmp;
            }
            j=j+1;
        }
        i=i+1;
    }
}

double sumList(int list[], int num, int drop)
{
    int i=drop;
    double sum=0;
    assert(num>drop);

    sort(list,num);

    while(i<num)
    {
        sum=sum+list[i];
        i=i+1;
    }
    return(sum);
}

double contrib(int list[], int list_max, int drop, int valueEach, int weight)
{
    int maxPossible;
    double earned;

    earned=sumList(list, list_max,drop);
    maxPossible=valueEach*(list_max-drop);

    return((earned/maxPossible)*weight);
}

// *********************************************************
// getScores()
//
//    Reads grades from a file.  We are assuming all the
//    grades are in order and are valid values.
// *********************************************************
Grades getScores()
{
    Grades in;
    int i=0;
    ifstream scoreFile;

    scoreFile.open("grades.txt");
    if (!scoreFile) {
        cout << "Unable to open file";
        exit(1); // terminate with error
    }


    // This too is a bit of a mess.  Ideally we'd read the file
    // and ignore comments and the like.  Further, the "cut-and-paste"
    // nature of this code is offensive.  But I don't have a better idea.
    i=0;
    while(i<NUM_EXAM)
    {
        scoreFile >>in.exam[i];
        i=i+1;
    }
    i=0;
    while(i<NUM_HW)
    {
        scoreFile >>in.hw[i];
        i=i+1;
    }
    i=0;
    while(i<NUM_INLAB)
    {
```

```
        scoreFile >>in.inlab[i];
        i=i+1;
    }
    i=0;
    while(i<NUM_PROJECT)
    {
        scoreFile >>in.project[i];
        i=i+1;
    }
    i=0;
    while(i<NUM_PRACTICAL)
    {
        scoreFile >>in.practical[i];
        i=i+1;
    }
    i=0;
    while(i<NUM_QUIZ)
    {
        scoreFile >>in.quiz[i];
        i=i+1;
    }
    cout << endl;
    return(in);
}
// Weight things so best exam worth 4/3 as much as rest
// For us that is 20% for best, 15% for rest.
double examValue(int exam[], int max)
{
    double all=sumList(exam,max,0);
    double best=sumList(exam,max,max-1);
    double rest=all-best;
    return(best*4.0/3.0+rest);
}
double projectContribution(int proj[], int max)
{
    double sum=0.0;

    // Because each project is worth a different amount,
    // there is no "generic" way to code this without being silly.
    // So we will check to make sure that the relivant project numbers
    // are consistant with what we are hard-coding here.
    assert(max==NUM_PROJECT);
    assert(27==WEIGHT_PROJECT);

    // Now just sum them.
    sum=sum+proj[0]*0.01;
    sum=sum+proj[1]*0.02;
    sum=sum+proj[2]*0.04;
    sum=sum+proj[3]*0.06;
    sum=sum+proj[4]*0.07;
    sum=sum+proj[5]*0.07;

    return(sum);
}



main()
{
    Grades courseGrades;
    double total=0.0;
    double tmp, best, sum;

    courseGrades=getScores();

    double examContrib, hwContrib, inlabContrib;
    double projectContrib, practicalContrib, quizContrib;

    // Homework -- Drop 1
    hwContrib=contrib(courseGrades.hw, NUM_HW ,1, OUTOF_HW, WEIGHT_HW);

    // Quiz -- Drop 0
    quizContrib=contrib(courseGrades.quiz, NUM_QUIZ ,0,
                        OUTOF_QUIZ, WEIGHT_QUIZ);
    // Practial -- Drop 0
    practicalContrib=contrib(courseGrades.practical, NUM_PRACTICAL,0,
                             OUTOF_PRACTICAL, WEIGHT_PRACTICAL);

    // Inlab -- Drop 2
    inlabContrib=contrib(courseGrades.inlab, NUM_INLAB ,2, OUTOF_INLAB,
                             WEIGHT_INLAB);

    // Project -- Special code
    //   Note: sorting Project scores would be BAD!
    projectContrib=projectContribution(courseGrades.project, NUM_PROJECT);

    // Exams
    //   We want best score multiplied by (4/3).  (So it is 20% rest are
    //   15%)  We use sumList to find the best.
    sum=sumList(courseGrades.exam, NUM_EXAM, 0);
    best=sumList(courseGrades.exam, NUM_EXAM, NUM_EXAM-1); // Best exam score.
    tmp=sum+(1.0/3.0)*best;
    examContrib=tmp/((NUM_EXAM+1.0/3.0)*OUTOF_EXAM)*WEIGHT_EXAM;

    // Print out the scores
    cout << "exams: " << examContrib  << endl;
    cout << "homework assignments: " << hwContrib << endl ;
    cout << "quizzes: " << quizContrib << endl ;
    cout << "inlabs " << inlabContrib << endl ;
    cout << "practicals: " << practicalContrib << endl ;
    cout << "projects: " << projectContrib << endl ;

    total=examContrib+hwContrib+quizContrib+practicalContrib+
          projectContrib+inlabContrib;
    cout << endl << "  TOTAL: " << total << endl;
}
```