Project Description:

For this project, you will be writing a program to calculate final grades in this class from raw assignment scores. There are a number of different functions that the test cases will call, however, you should be writing additional helper functions to aid in clarity and to encapsulate shared functionality (thus reducing duplicated code).

25 points of this project is only available if avoid using looping statements (`for` , `while` , `for_each`) and instead utilize the STL provided algorithms. Using other approaches (like recursion) will result in the loss of these points. I recommend using an algorithm first approach: for each problem you encounter, think first about what algorithm could be useful here, instead of trying to write a traditional loop and trying to refactor it later.

> ★ I used many different algorithms in my solution, but the most common one was `transform` (used 7 times).

> ⚠ Note, that this project encourages you to avoid using looping statements even when loops would be an objectively better choice. We are asking for you to demonstrate a strong proficiency with the STL algorithms at the expense of clarity.

Your project should be separated into only two files, "main.h" and "main.cpp". Please write legible code, as we will be grader more severely than on Project 1.

Required Functions:

- The function, `GetPointTotalForStudent` , returns an int representing the number of points earned in total on a particular type of assignment. As parameters it takes a map of strings to strings (which I will call student_info) and a second parameter of a string. The map's keys are the names of assignments and the values are the scores earned on each assignment. The map also contains other data that isn't relevant to this function. The second parameter is a string specifying the category of assignment.
  For example, if the category is "Exam". All the scores of assignments with "Exam" in the name should be tallied.
- The function, `GetTopNHomeworkTotalForStudent` , takes 2 parameters, the first being the same map named student_info as in the previous function. The second parameter is an int representing the number of homework assignments to tally up. Remember that only the top homework scores contribute to the tally.
- The function, `GetNumberOfMissingLabsForStudent` , takes only student_info, and it returns the number of labs that don't have a score of exactly "1".
- The function, `GetPointTotalForStudent` , takes only student_info, and returns the number of points earned by that student. Be sure to consult the syllabus (https://cse232-msu.github.io/CSE232/syllabus.html) as to what scores contribute to the total.
- The function, `GetIDToInfoFromCSV` , takes a single parameter a string denoting a filename for a CSV formatted file. This function returns a map of string (the ID) to student_info (as mentioned previously a map of names to values/scores). This return value will be called id_to_student_info henceforth.
  > ★ Although there are many ways to tackle this function, my solution used (among other functions) `std::generate` .
- The function, `GetIDToGrade` , takes a id_to_student_info map and returns a map of string (ID) to double (the student's calculated grade). Be sure to account for the missed lab penalty. Also note the "Project Honors" isn't worth any points that contribute to the grade calculation.
- The function, `GetStudentsEligibleForHonorsCredit` , takes a id_to_student_info map and an int (the minimum required grade on the Project Honors) and returns a set of strings (IDs) of the students that meet the honors requirements. Please also see the syllabus for the minimum final grade requirement as well.

```
#include "main.h"

// This function is what I use to to sort different vectors.
bool cmp(pair<string, int>& a,
         pair<string, int>& b)
{
    return a.second > b.second;
}
// In this function, what I use is a set and start adding to it if student info came
back with a good grade, and if it came higher than project Honors, also if it came
back higher than the gpa required then have it put into my set.
```

```
std::set<std::string>
GetStudentsEligibleForHonorsCredit(map<string,map<string,string>> student_info, int
a){
    std::set<string> IDHonors;
    map<string,double> Grades = GetIDToGrade(student_info);

std::transform(student_info.begin(),student_info.end(),Grades.begin(),std::inserter(I
DHonors,IDHonors.end()),[&a](auto student_info,auto m){
        int finals = GetPointTotalForStudent(student_info.second,"Project Honors");
        if (finals >=a)
        {
            if (m.second>=3.5)
            {
                return student_info.first;
            }

        }
        string empty = "";
        return empty;
    });
    std::set<string> IDHonorsFinal;

std::copy_if(IDHonors.begin(),IDHonors.end(),std::inserter(IDHonorsFinal,IDHonorsFina
l.end()),[](auto a){
        if (a.empty())
        {
            return false;
        }
        return true;
    });
    return IDHonorsFinal;
}
// This checks the key fo r the string Lab, and If it is found then dont add to the
result, otherwise you should add to result
int GetNumberOfMissingLabsForStudent(map<string,string> student_info){
    int result = std::accumulate(student_info.begin(),student_info.end(),0,[](int a1,
pair<string,string> a){
        string firstpair = a.first;
        string secondpair = a.second;
        std::size_t found = firstpair.find("Lab");
        if (found!=std::string::npos)
        {
            if (secondpair.empty())
            {return a1 +1;}
            if (secondpair.find_first_not_of("0123456789")== string::npos)
            {int secondint = stoi(secondpair);
            if (secondint == (1))
                {return a1;}
            }
            return a1 + 1;
        }
        return a1;
    });
    return result;
}
```

```
// IN this function I look through each key fo r the key and if it is found. Then
check if there is a number inside of it, otherwise dont add that value
int GetPointTotalForStudent(map<string,string> student_info, string keyword){
    int result =
std::accumulate(student_info.begin(),student_info.end(),0,[keyword](int a1,
pair<string,string> a){
        string firstpair = a.first;
        string secondpair = a.second;
        std::size_t found = firstpair.find(keyword);
        if (found!=std::string::npos)
        {
            if (secondpair.empty())
            {return a1;}
            if (secondpair.find_first_not_of("0123456789")== string::npos)
            {int secondint = stoi(secondpair);
            return secondint + a1;}
        }
        return a1;
    });
    return result;
}
// in this function I first make a map and add the homework values only, then I go
through those values and change them to a integer. I make a vector that I use to sort
the vector according to the highest scores. after that I get the amount that is
needed to get the top homeworks
int GetTopNHomeworkTotalForStudent(map<string,string> student_info, int keyword){
    map<string,string> homeworks = {};

std::copy_if(student_info.begin(),student_info.end(),std::inserter(homeworks,homework
s.end()),[](pair<string,string> a){
        string firstpair = a.first;
        string secondpair = a.second;
        string homework = "HW";
        std::size_t found = firstpair.find(homework);
        if (found != std::string::npos)
        {
            if (secondpair.empty())
            {return false;}
            if (secondpair.find_first_not_of("0123456789")== string::npos)
            {return true;}
        }
        return false;
    });
    map<string,int> homework_int;

std::transform(homeworks.begin(),homeworks.end(),std::inserter(homework_int,homework_
int.end()),
    [](pair<string,string> a){
        pair<string,int> b = {"HW",0};
        string firstpair = a.first;
        string secondpair = a.second;
        int secondint = stoi(secondpair);
        b.first = a.first;
        b.second = secondint;
        return b;
```

```
    });
    vector<pair<string, int> > A;

    copy(homework_int.begin(),homework_int.end(),back_inserter(A));

    sort(A.begin(), A.end(), cmp);
    auto size_vec = A.size();
    size_vec = static_cast<int>(size_vec);
    int value;
    if (size_vec<keyword)
    {
        value = size_vec;
    }
    else
    {value = keyword;}
    int accumulate_final =std::accumulate(A.begin(),A.begin()+(value),0,[](int
a1,pair<string,int> a2){
        return a1 + a2.second;
    });
    return accumulate_final;
}
// in this function I have it open up the file, I then iterate through the first line
and make a key of Vectors. Also have empty vectors there, and then check get rid of
the empty strings. I iterate through the csv file and check fo r a comma, if the
comma is found then add the value to the vector. At the end I add that vector to a
map  that will be student info.
map<string,map<string,string>> GetIDToInfoFromCSV(string a){
    std::ifstream input;
    input.open(a);
    string ss;
    getline(input,ss);
    string s33;
    ss.push_back(',');
    vector<pair<string,string>> Keys;
    std::transform(ss.begin(),ss.end(),std::back_inserter(Keys),[&s33](auto c){
        if (c == ',')
        {
            pair<string,string> news= {s33," "};
            s33.clear();
            return news;
        }

        s33.push_back(c);
        pair<string,string> d = {"",""};
        return d;
    });
    vector<pair<string,string>> Keys_Final;

std::copy_if(Keys.begin(),Keys.end(),std::back_inserter(Keys_Final),[](pair<string,st
ring> a){
        string firstval = a.first;
        if (firstval.empty())
        {
            return false;
        }
```

```
        return true;
    });
    map<string,string> random;
    map<string,map<string,string>> final;
    string s22;
    int count =0;

std::transform(std::istreambuf_iterator<char>(input),std::istreambuf_iterator<char>()
,std::inserter(random,random.end()),[&s22,&Keys_Final,&count,&final](auto c){
        if (c== ',' && s22.empty())
        {
            Keys_Final.at(count).second = s22;
            s22.clear();
            count++;
            pair<string,string> bb;
            return bb;
        }

        if (c == ',')
        {
            Keys_Final.at(count).second = s22;
            s22.clear();
            count++;
            pair<string,string> bb;
            return bb;
        }
        if (c == '\n')
        {
            Keys_Final.at(count).second = s22;
            s22.clear();
            map<string,string> finalsss;

copy(Keys_Final.begin(),Keys_Final.end(),std::inserter(finalsss,finalsss.end()));
            pair <string,map<string,string>> f = {Keys_Final.at(1).second,finalsss};
            final.insert(f);
            count = 0;
            pair<string,string> bb;
            return bb;
        }
        s22.push_back(c);
        pair<string,string> bb;
        return bb;
    });
    Keys_Final.at(count).second = s22;
    map<string,string> finalsss;
    copy(Keys_Final.begin(),Keys_Final.end(),std::inserter(finalsss,finalsss.end()));
    pair <string,map<string,string>> f = {Keys_Final.at(1).second,finalsss};
    final.insert(f);
    return final;
}
// This will be used by using student info, Iterate through it and I add to a map
that is a string an and it will add the values once it does the math
map<string,double> GetIDToGrade(map<string,map<string,string>> student_info){
    map<string,double> student_info_final;
```

```
std::transform(student_info.begin(),student_info.end(),std::inserter(student_info_fin
al,student_info_final.end()),[&student_info_final](auto a){
        string firstpair = a.first;
        map<string,string> secondpair = a.second;
        pair<string,double> h;
        int finalpoints = GetPointTotalForStudent(secondpair);
        int missinglabs = GetNumberOfMissingLabsForStudent(secondpair);
        if (missinglabs >=2)
        {
            missinglabs = missinglabs -2;
        }
        else
        {
            missinglabs = 0;
        }

        float missinglabss = missinglabs *.5;

        map<int,float> scores =
{{1000,4.0},{950,4.0},{900,4.0},{850,3.5},{800,3.0},{750,2.5},{700,2.0},{650,1.5},{60
0,1.0}};
        int finalgrade = finalpoints %50;
        finalpoints = finalpoints - finalgrade;
        double finalgradess;
        if (finalpoints <600)
        {
            finalgradess = 0.0;
        }
        else
        {
            finalgradess = scores.at(finalpoints);
        }
        finalgradess = finalgradess - missinglabss;
        if (finalgradess <=0.0)
        {
            finalgradess =0.0;
        }
        h.first = firstpair;
        h.second = finalgradess;
        return h;
    });
    return student_info_final;
}
// Very simple function that just adds the values f or everything
int GetPointTotalForStudent(map<string,string> student_info){
    int examgrades = GetPointTotalForStudent(student_info, "Exam");
    int projectgrades = GetPointTotalForStudent(student_info, "Project 1");
    int projectgrades2 = GetPointTotalForStudent(student_info,"Project 2");
    int projectgrades3 = GetPointTotalForStudent(student_info,"Project 3");
    int homework = GetTopNHomeworkTotalForStudent(student_info, 15);
    return examgrades +projectgrades+projectgrades2+projectgrades3+ homework;
    }

int main(){
```

```
}
```