```
// Written by Jared Dyreson
// Days bug free: -1
#include <iostream>
#include <string>
#include <fstream>
#include <algorithm>
#include <sstream>
using namespace std;

// https://stackoverflow.com/questions/14548753/passing-a-multidimensional-variable-length-arr
ay-to-a-function
// passing 2-D array like <int (*scores)[6]> . Which looks cleaner than <int scores[][6]> in m
y opinion

// all my prototypes, in alphabetical order
char getInput();
char returnLetterGrade(int percentage);
int calculatePercentGrade(double scoreOfStudent);
int removeDuplicates(string array[], int SIZE);
void clear();
void findHighestGrade(int NAMES, int GRADES, int (*scores)[6], string* names);
void findStudent(string* names, int (*scores)[6], const int numberOfScores, const int numberOf
Students);
void printAllInformation(int (*scores)[6], string* names, char* letterGrades, const int number
OfScores, const int numberOfStudents);
void printMenu();
void saveInformation(string* names, char* grades, int (*scores)[6], const int numberOfStudents
, const int numberOfScores);
void sendToArray(ifstream &stream, int studentSize, const int numberOfScores, int (*scores)[6]
, string* names, char* letterGrades, string* catchAll);

int main(){

  const int numberOfScores = 6;
  const int numberOfStudents = 23;

  ifstream stream;
  stream.open("/home/jared/Desktop/CompSci-CSUF/CPSC-121/Projects/Project Two/CPSC121dataUpdat
ed.txt");
  if (!stream.is_open()){
    cerr << "Could not open file, cowardly refusing" << endl;
    exit(1);
  }

  // send all data to array named catchAll, find duplicates from there and parse non duplicate
s to respective arrays
  // much easier than trying to parse out the duplicate lines like "cat CPSC121dataUpdated.txt
 | uniq | sort - n" which was the first idea but I quickly realized I was in over my head afte
r reading -> https://opensource.apple.com/source/text_cmds/text_cmds-71/uniq/uniq.c.auto.html
  char letterGrades[numberOfStudents] = {};
  char selection;
  int scores[numberOfStudents][numberOfScores] = {0};
  int totalScores[numberOfStudents] = {0};
  string catchAll[numberOfStudents] = {};
  string names[numberOfStudents] = {};

  sendToArray(stream, numberOfStudents, numberOfScores, scores, names, letterGrades, catchAll)
;
  // my cleanest switch case in a long time
  do
  {
    printMenu();
    cin >> selection;
    switch (selection)
    {
      case 'A':
      {
        clear();
        printAllInformation(scores, names, letterGrades, numberOfScores, (numberOfStudents - 1
));
        break;
      }
      case 'H':
      {
        clear();
        findHighestGrade(numberOfStudents, numberOfScores, scores, names);
        break;
      }
      case 'S':
      {
        clear();
        findStudent(names, scores, numberOfScores, numberOfStudents);
        break;
      }
      case 'G':
      {
        clear();
        saveInformation(names, letterGrades, scores, numberOfStudents, numberOfScores);
        break;
      }
      case 'Q':
      {
        break;
      }
    }
  } while (selection != 'Q');
  return 0;
}

void printAllInformation(int (*scores)[6], string* names, char* letterGrades, const int number
OfScores, int trimmedAmount){
  for (int i = 0; i < trimmedAmount; i++){
    cout << "Student: " << names[i] << endl;
    cout << "Grade: " << letterGrades[i] << endl;
    for (int j = 0; j < numberOfScores; j++){
      // formatting
      if (j == (numberOfScores - 1)){
        cout << scores[i][j] << endl;
      }
      else{
        cout << scores[i][j] << " ";
      }
    }
  }
}

int removeDuplicates(string array[], int SIZE){
  // index of the previous element in the array we are trying to compare
  int prev = 0;
  // loop through all the elements
  for (int i = 0; i < SIZE; i++){
    // if the current element is not equal to the previous element
    if (array[i] != array[prev]){
      // increment the previous counter to keep up with the loop
      array[++prev] = array[i];
    }
  }
  int count = prev + 1;
  // return the amount of elements that are not duplicates
  return count;
```

```
}

void printMenu(){
  // display options and selection prompt
  cout << "+------+" << endl;
  cout << "| Menu |" << endl;
  cout << "+ -----+" << endl;
  cout << "A) Display all information about students" << endl;
  cout << "H) Highest grade in the class" << endl;
  cout << "S) Search for a student" << endl;
  cout << "G) Save Student information to StudentData.txt" << endl;
  cout << "Q) Quit" << endl;
  cout << "Selection: ";
}

char returnLetterGrade(int percentage){
  // we can simplify indexing the grade for every student with a function and return the char
at the end
  char grade;
  if (percentage == 100 && percentage < 100 && percentage >= 90){
    grade = 'A';
  }
  else if (percentage <= 89 && percentage >= 80){
    grade = 'B';
  }
  else if (percentage <= 79 && percentage >= 70){
    grade = 'C';
  }
  else if (percentage <= 69 && percentage >= 60){
    grade = 'D';
  }
  else {
    grade = 'F';
  }
  return grade;
}

void sendToArray(ifstream &stream, int studentSize, const int numberOfScores, int (*scores)[6]
, string* names, char* letterGrades, string* catchAll){
  string line;
  // I am aware that there are two student sizes under different aliases but I wanted to keep
the parameters for sendToArray as low as possible as
  int counter = 0, number, sum = 0;
  while(getline(stream, line)){
    catchAll[counter] = line;
    counter++;
  }
  sort(catchAll, catchAll+studentSize);
  int count = removeDuplicates(catchAll, studentSize);
  string firstName, lastName, fullName;
  stream.close();
  for (int i = 0; i < count; i++){
    string var;
    // this was a major breakthrough. In project one, we already had a ifstream available at o
ur disposal and in this project we needed to recreate it
    // once we recreate it, we treat the array that stores our parsed name list as a file in r
unning memory, allowing to eliminate the need to write and open a separate file
    istringstream ss(catchAll[i]);
    getline(ss, line, ',');
    // now we grab the name
    stringstream namesStream(line);
    while (namesStream  >> firstName >> lastName){
      fullName = firstName + " " + lastName;
      // send it to the array
      names[i] = fullName;
    }
    for (int j = 0; j < numberOfScores; j++){
      getline(ss, line, ',');
      stringstream scoresStream(line);
      //replaces stoi <poor man's version or pre c++11>
      // converting string to int
      while (scoresStream >> number){
        scores[i][j] = number;
      }
    }
  }
  // prevents the need for another array, hence more memory efficient
  // calculating it on the fly
  for (int i = 0; i < (studentSize - 1); i++){
    for (int j = 0; j < numberOfScores; j++){
      sum+=scores[i][j];
    }
    // since this is a part of the for loop, we can call the returnLetterGrade function to ass
ign the correct letter grade and send it to the array
    char grade = returnLetterGrade(sum);
    letterGrades[i] = grade;
    // reset the counter
    sum = 0;
  }
}

void clear(){
  // this was not created by me -> https://stackoverflow.com/questions/4062045/clearing-termin
al-in-linux-with-c-code
  // works just like /usr/bin/clear !
  cout << "\033[2J\033[1;1H";
}

void findStudent(string* names, int (*scores)[6], const int numberOfScores, const int numberOf
Students){
  string studentSelection;
  cout << "Enter name: ";
  cin.ignore();
  getline(cin, studentSelection);
  bool studentIsFound;
  for (int i = 0; i < numberOfStudents; i++) {
    // we found them
    if (names[i] == studentSelection){
      cout << studentSelection << " is in our records" << endl;
      // this is so we can check after this loop ends if the student is not found
      studentIsFound = true;
      for (int k = 0; k < numberOfScores; k++){
        // display all of the scores
        // when we reach the end of the loop, we change the formatting, printing a newline and
 allowing the rest of the menu to be displayed
        if (k == (numberOfScores - 1)){
          cout << scores[i][k] << endl;
        }
        else{
          // print all in one line while it is not the last
          cout << scores[i][k] << " ";
        }
      }
      // no need to continue going through the loop since we found them
      break;
    }
    else{
      // continue to say that we cannot find the student
      studentIsFound = false;
    }
  }
  // this is where having that extra boolean assignment comes in handy
  // if we did find the student, this condition is false and will not be executed
  // this will allow us not to prematurely spout out that we could not find the student becaus
e a certain index returned false
  if (!studentIsFound) {
    cerr << "We could not find " << studentSelection << " in our records" << endl;
  }
}

void saveInformation(string* names, char* grades, int (*scores)[6], const int numberOfStudents
, const int numberOfScores){
  cout << "Saving...." << endl;
  // open a new file to write to
  ofstream write;
  write.open("StudentData.txt");
  for (int i = 0; i < (numberOfStudents - 1); i++){
    // all names and grades
    write << "Student: " << names[i] << endl;
    write << "Letter Grade: " << grades[i] << endl;
    for (int j = 0; j < numberOfScores; j++){
      // scores
      if (j == (numberOfScores - 1)){
        write << scores[i][j] << endl;
        write << "\n";
      }
      else{
        write << scores[i][j] << " ";
      }
    }
  }
  write.close();
  cout << "Successfully wrote data to StudentData.txt" << endl;
}

void findHighestGrade(const int NAMES, const int GRADES, int (*scores)[6], string* names){
  // these are temporary varibales used for comparision
  int counter = 0, largest = 0;
  // store the student witht the highest grade in this string
  string studentName;
  for (int i = 0; i < NAMES; i++){
    for (int j = 0; j < GRADES; j++){
      // calculate the score for a given student
      counter+=scores[i][j];
      }
      // if your sum is bigger than the largest, obviously should be denoted as largest
      // since it is in a loop, it can be overriden for a new student
    if(counter > largest){
      largest = counter;
      studentName = names[i];
    }
    // reset the counter
    counter = 0;
  }
  cout << "Student with the Highest Score: " << studentName << endl;
  cout << "Grade: " << largest << "%" << endl;
}
```

Terminal
File Edit View Search Terminal Help

```
Student: Barbara Tran
Grade: C
3 14 20 12 18 11
Student: Bella Morrison
Grade: C
4 12 18 13 18 14
Student: Bill Benson
Grade: D
5 14 10 15 13 12
Student: Chris Smith
Grade: C
5 17 25 10 8 14
Student: Dave Ramirez
Grade: C
3 14 20 12 18 11
Student: Dennis Bensen
Grade: C
4 12 18 13 18 14
Student: Eva Simon
Grade: D
5 10 15 10 11 14
Student: Jen Doe
Grade: D
2 11 13 13 10 13
Student: Jim Lin
Grade: F
1 0 11 15 17 12
Student: John Connell
Grade: C
3 10 24 10 11 14
Student: John Smith
Grade: D
5 10 15 10 11 14
Student: John Thomson
Grade: C
5 10 14 15 17 12
Student: Ken Norris
Grade: B
5 15 18 10 11 24
Student: Kim White
Grade: B
5 10 21 15 17 12
Student: Larry Tran
Grade: C
3 14 20 12 18 11
Student: Linda Lukas
Grade: B
4 10 13 25 10 23
Student: Mary Doe
Grade: B
4 15 13 25 10 13
Student: Phil Moon
Grade: B
4 12 18 13 18 16
Student: Randy Anderson
Grade: C
4 12 18 10 18 14
Student: Tim Baker
Grade: C
4 12 18 13 18 14
Student: Tom Johnson
Grade: C
5 10 11 15 17 12
Student: Warren Doe
Grade: B
4 15 13 25 10 17
```

Terminal
File
Edit
View
Search
Terminal
Help

```
Student with the Highest Score: Linda Lukas
Grade: 85%
+------+
| Menu |
+ -----+
A) Display all information about students
H) Highest grade in the class
S) Search for a student
G) Save Student information to StudentData.txt
Q) Quit
Selection: 
```

Terminal
File Edit View Search Terminal Help

```
Enter name: Jared Dyreson
We could not find Jared Dyreson in our records
+------+
| Menu |
+ -----+
A) Display all information about students
H) Highest grade in the class
S) Search for a student
G) Save Student information to StudentData.txt
Q) Quit
Selection: 
```

Terminal
File Edit View Search Terminal Help

```
Enter name: Linda Lukas
Linda Lukas is in our records
4 10 13 25 10 23
+------+
| Menu |
+ -----+
A) Display all information about students
H) Highest grade in the class
S) Search for a student
G) Save Student information to StudentData.txt
Q) Quit
Selection: 
```

Terminal
File Edit View Search Terminal Help

```
Saving....
Successfully wrote data to StudentData.txt
+------+
| Menu |
+ -----+
A) Display all information about students
H) Highest grade in the class
S) Search for a student
G) Save Student information to StudentData.txt
Q) Quit
Selection: 
```

jared@jared-xps ~/Desktop/CompSci-CSUF/CPSC-121/Projects/Project Two
File Edit View Search Terminal Help

```
jared@jared-xps ~/Desktop/CompSci-CSUF/CPSC-121/Projects/Project Two $ cat StudentData.txt
Student: Barbara Tran
Letter Grade: C
3 14 20 12 18 11

Student: Bella Morrison
Letter Grade: C
4 12 18 13 18 14

Student: Bill Benson
Letter Grade: D
5 14 10 15 13 12

Student: Chris Smith
Letter Grade: C
5 17 25 10 8 14

Student: Dave Ramirez
Letter Grade: C
3 14 20 12 18 11

Student: Dennis Bensen
Letter Grade: C
4 12 18 13 18 14

Student: Eva Simon
Letter Grade: D
5 10 15 10 11 14

Student: Jen Doe
Letter Grade: D
2 11 13 13 10 13

Student: Jim Lin
Letter Grade: F
1 0 11 15 17 12

Student: John Connell
Letter Grade: C
3 10 24 10 11 14

Student: John Smith
Letter Grade: D
5 10 15 10 11 14

Student: John Thomson
Letter Grade: C
5 10 14 15 17 12

Student: Ken Norris
Letter Grade: B
5 15 18 10 11 24

Student: Kim White
Letter Grade: B
5 10 21 15 17 12

Student: Larry Tran
Letter Grade: C
3 14 20 12 18 11

Student: Linda Lukas
Letter Grade: B
4 10 13 25 10 23

Student: Mary Doe
```

```
Saving....
Successfully wrote data to StudentData.txt
+------+
| Menu |
+ -----+
A) Display all information about students
H) Highest grade in the class
S) Search for a student
G) Save Student information to StudentData.txt
Q) Quit
Selection: Q
Press any key to continue...
```