```
// Written by Jared Dyreson
// File: Student.hpp
// Initialize Student class with given attributes and functions
#pragma once
#include <string>
using namespace std;
class Student{
  private:
    string name;
    int score, amount;
    // huh, put the array as a mutable object inside the class attributes....
    // if I didn't, I would then need to Initialize it inside of main in Source.cpp and comple
tely defeats the purpose!
    string *names;
    int *scores;
  public:
    void setScore(int s);
    int getScore();
    void setName(string n);
    string getName();
    Student();
    ~Student();
};
```

```
// Written by Jared Dyreson
// File: Student.cpp
// Define all of the functions with their use-cases

#include "Student.hpp"
#include <iostream>
#include <string>
using namespace std;

Student:: Student(){}
string Student:: getName(){ return name; }
void Student:: setName(string n){
  name = n;
}
int Student:: getScore() { return score; }
void Student:: setScore(int s){
  score = s;
}
Student:: ~Student(){}
```

```
// Written by Jared Dyreson
// File : Source.cpp
// Main implementation

/*
  Pseudocode begin
  - grab amount of students in classroom <numberOfStudents>
  - construct a super student object that represents all of the students a one object <loadArr
ays>
    - that student object creates the arrays needed while assigning values to the correct arra
ys
  - calulate the average of all the students
  - scores array is then used to dermine which range gets a star and how many of them get that
 mythical star<displayData>
  - display all of the data not including their names as we are not required to do so <display
Data>
  - properly delete estudiante from the heap
    - we cannot let our little astronaut detatch from the space shuttle again
      - no one asked for a Gravity 2....
  - assign null pointer to the object so it is properly dealt with
  Pseudocode end
*/

#include "Student.hpp"
#include <iostream>
#include <string>
using namespace std;

Student * loadArrays(int size);
void displayData(Student *myStudents, const int size);
int numberOfStudents();
double calculateAverage(Student *myStudents, const int size);
void clear();

int main(){
  int amount = numberOfStudents();
  Student *estudiante = loadArrays(amount);
  displayData(estudiante, amount);
  delete [] estudiante;
  estudiante = nullptr;
  return 0;
}

Student * loadArrays(int size){
  // create a new object pointing to the dynamic array
  Student *myStudent = new Student[size];
  // temporary variables
  int input;
  string name;
  // loop through as many times needed
  for(int i = 0; i < size; i++){
    // their grade
    cout << "Grade: ";
    cin >> input;
    // their name
    cout << "Name: ";
    cin.ignore();
    getline(cin, name);
    // set the name and score via public method to a private member in the Student class
    myStudent[i].setScore(input);
    myStudent[i].setName(name);
  }
  return myStudent;
}
```

```
void displayData(Student *myStudents, const int size){
  clear();
  // grab the class average
  cout << "Class Average: " << calculateAverage(myStudents, size) << endl;
  // number of students...redundant as RAID 0
  cout << "Number of Students: " << size << endl;
  // ooh pretty
  cout << "+--------------------------+" << endl;
  cout << "| Total Grade Distribution |" << endl;
  cout << "+--------------------------+" << endl;
  // this is where the power of an algorithm comes into play
  const int number = 11;
  // we see that the grades have a range of 9 so we're gonna take advantage of it
  int compareMe[number] = {0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100};
  for(int i = 0; i < number; i++){
    // only part where this fails so we're making a hard-coded check
    if(compareMe[i] == 100){
      cout << compareMe[i];
    }
    else{
      // everything else gets the base number and base number plus 9 to emulate a range
      cout << compareMe[i] << " - " << (compareMe[i] + 9) << ":";
    }
    for(int j = 0; j < size; j++){
      // if the current score for the student is inside the range, simply print a '*'
      if(myStudents[j].getScore() <= (compareMe[i] + 9) && myStudents[j].getScore() >= compare
Me[i]){
        cout << " *";
    }
  }
  cout << endl;
  }
}
int numberOfStudents(){
  // grab the current amount of students
  int amount;
  cout << "How many students?: ";
  cin >> amount;
  return amount;
}

double calculateAverage(Student *myStudents, const int size){
  double sum = 0.0;
  for(int i = 0;i < size; i++){
    // traverse the array, adding ontop of the sum
    int particularStudentScore = myStudents[i].getScore();
    sum+=particularStudentScore;
    // reset with null byte so nothing is overidden
    particularStudentScore = '\0';
  }
  // allow us to use this value outside of the function
  return (sum / size);
}

// clear the screen
void clear(){ cout << "\033[2J\033[1;1H"; }
```

File Edit View Search Terminal Help

```
jared@jared-xps ~/Desktop/CompSci-CSUF/CPSC-121/Projects/Project Four $ ./Project_Four
How many students?: 3
Grade: 85
Name: Jared Dyreson
Grade: 97
Name: Gabriel Ball
Grade: 100
Name: Levi Randall
```

```
Class Average: 94
Number of Students: 3
+--------------------------+
| Total Grade Distribution |
+--------------------------+
0 - 9:
10 - 19:
20 - 29:
30 - 39:
40 - 49:
50 - 59:
60 - 69:
70 - 79:
80 - 89: *
90 - 99: *
100 *
jared@jared-xps ~/Desktop/CompSci-CSUF/CPSC-121/Projects/Project Four $
```