// Solution 6: Storing test grades in 1-d array
// using run-time array allocation

#include <iostream>     // cin, cout
#include <cstdlib>      // exit()
#include <iomanip>      // setw(), setprecision()
#include <cstring>      // strlen()

using namespace std;

struct Student       // External Structure Template
{
    char* pname;
    int grade;
};

void get_maximum_students(int& max_students);
Student* allocate_array(int max_students);
void get_students(Student* pStu, int max_students, int& student_count);
void show_students(const Student* pStu, int student_count);
void show_class_average(const Student* pStu, int student_count);
void deallocate_array(Student* pStu, int student_count);

int main()
{
    int maximum_students;
    get_maximum_students(maximum_students);
    Student* pStudent = allocate_array(maximum_students);  // Memory was allocated successfully
    int student_count;
    get_students(pStudent, maximum_students, student_count);
    show_students(pStudent, student_count);
    show_class_average(pStudent, student_count);
    deallocate_array(pStudent, student_count);
    return 0;
}

void get_maximum_students(int& max_students)
{
    cout << "Enter the maximum number of students==>";
    cin >> max_students;
    if (max_students <= 0)
    {
        cerr << "\nMaximum number of students must be positive.\n";
        exit(EXIT_FAILURE);
    }
}
Student* allocate_array(int max_students)
{
    Student* pStu = new Student[max_students]; // Allocate memory for Students
    if (!pStu) // Did new fail?
    {
        cerr << "Not able to allocate memory for the array\n";
        exit(EXIT_FAILURE);
    }
    return pStu;
}

void get_students(Student* pStu, int max_students, int& student_count)
{
    cout << "Enter a maximum of " << max_students << " students.\n" << "Enter the name of the first student (<Enter> to QUIT): ";
    char temp_name[80];
    int i = 0;
    cin.ignore(1024, '\n'); // flush cin stream
    while (i < max_students && cin.getline(temp_name, 80) && temp_name[0])
    {
        (pStu + i)->pname = new char[strlen(temp_name) + 1];
        if (!(pStu + i)->pname) // Test new
        {
            cerr << "Not able to allocate memory for the student's name\n";
            for (int j = 0; j < i; j++)
                delete [] (pStu + j)->pname; // Deallocate names of students
            delete [] pStu; // Deallocate memory for the Students
            exit(EXIT_FAILURE);
        }
        strcpy((pStu + i)->pname, temp_name);
        cout << "Enter student's grade: ";
        cin >> (pStu + i)->grade;
        i++;
        cin.get(); // Flush the newline
        if (i < max_students)
            cout << "Enter the name of the next student (<Enter> to QUIT): ";
    }
    student_count = i; // Set the actual number of Students
    if (!student_count)
    {
        cerr << "\nYou entered no students.\n";
        delete [] pStu;
        exit(EXIT_FAILURE);
    }
}
void show_students(const Student* pStu, int student_count)  
{  
cout << "\nYour " << student_count << " student"  
<< (student_count > 1 ? "s are" : " is") << ":\n\n";  
for (int i = 0; i < student_count; i++) // Output the Students  
{  
cout << left << setw(20) << (*(pStu + i)).pname  
<< right << setw(6) << (*(pStu + i)).grade << '\n';  
}  
}  

void show_class_average(const Student* pStu, int student_count)  
{  
int grade_sum = 0;  
for (int i = 0; i < student_count; i++)  
grade_sum += pStu[i].grade; // Accumulate the grades  
cout << "\nClass average is: "  
<< fixed << setprecision(1)  
<< float (grade_sum) / student_count << "\n";  
}  

void deallocate_array(Student* pStu, int student_count)  
{  
for (int i = 0; i < student_count; i++)  
delete [] pStu[i].pname; // Deallocate the names of the students  
delete [] pStu; // Deallocate memory for the Students  
}  

**Run 1:**  
Enter the maximum number of students===>5  
Enter a maximum of 5 students.  
Enter the name of the first student (<Enter> to QUIT): *Joe E. Black*  
Enter student's grade: 75  
Enter the name of the next student (<Enter> to QUIT): *Pat Wong*  
Enter student's grade: 80  
Enter the name of the next student (<Enter> to QUIT): *Jim Garcia*  
Enter student's grade: 99  
Enter the name of the next student (<Enter> to QUIT): *Sue Tang*  
Enter student's grade: 100  
Enter the name of the next student (<Enter> to QUIT): *Sam Hadwan*  
Enter student's grade: 74  
Your 5 students are:  

<table>
<thead>
<tr>
<th>Name</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joe E. Black</td>
<td>75</td>
</tr>
<tr>
<td>Pat Wong</td>
<td>80</td>
</tr>
<tr>
<td>Jim Garcia</td>
<td>99</td>
</tr>
<tr>
<td>Sue Tang</td>
<td>100</td>
</tr>
<tr>
<td>Sam Hadwan</td>
<td>74</td>
</tr>
</tbody>
</table>

Class average is: 85.6
Run 2:
Enter the maximum number of students===>5
Enter a maximum of 5 students.
Enter the name of the first student (<Enter> to QUIT): Joe Edward Black
Enter student's grade: 75
Enter the name of the next student (<Enter> to QUIT): Pat Wong
Enter student's grade: 80
Enter the name of the next student (<Enter> to QUIT): James Garcia
Enter student's grade: 99
Enter the name of the next student (<Enter> to QUIT):
Your 3 students are:
Joe Edward Black 75
Pat Wong 80
James Garcia 99
Class average is: 84.7

Run 3:
Enter the maximum number of students===>1
Enter a maximum of 1 students.
Enter the name of the first student (<Enter> to QUIT): James F. Garcia
Enter student's grade: 100
Your 1 student is:
James F. Garcia 100
Class average is: 100.0

Run 4:
Enter the maximum number of students===>5
Enter a maximum of 5 students.
Enter the name of the first student (<Enter> to QUIT):
You entered no students.

Run 5:
Enter the maximum number of students===>0
Maximum number of students must be positive.
Dynamic Array for Run 2:

Syntax to access members

1. pStu[i].pname
2. (pStu + i)->pname
3. (*(pStu + i)).pname